ThinApp-Benutzerhandbuch

ThinApp 4.5

Dieses Dokument unterstützt die Version jedes hier aufgeführten Produkts und alle nachfolgenden Versionen, bis es durch eine neue Ausgabe ersetzt wird. Möglicherweise neuere Ausgaben dieses Dokuments finden Sie unter http://www.vmware.com/support/pubs.

DE-000219-00

vmware®

Die neueste technische Dokumentation finden Sie auf der Website von VMware unter:

http://www.vmware.com/support/

Die Website von VMware bietet Ihnen auch die neuesten Produkt-Updates.

Falls Sie Anmerkungen zu dieser Dokumentation haben, senden Sie diese bitte an:

docfeedback@vmware.com

Copyright © 2010 VMware, Inc. Alle Rechte vorbehalten. Dieses Produkt ist durch das US-amerikanische und internationale Urheberrecht und geistige Eigentumsrecht geschützt. Die Produkte von VMware sind durch ein oder mehrere Patente geschützt. Diese sind unter http://www.vmware.com/go/patents aufgeführt.

VMware ist eine eingetragene Marke oder Marke von VMware, Inc. in den USA und/oder anderen Ländern. Alle anderen in diesem Dokument erwähnten Bezeichnungen und Namen sind unter Umständen markenrechtlich geschützt.

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304, USA
www.vmware.com

Inhalt

Über dieses Handbuch 9

1 Installation von ThinApp 11
- ThinApp-Installationsanforderungen 11
  - Betriebssysteme, Anwendungen und Systeme, die von ThinApp unterstützt werden 11
  - Anwendungen, die von ThinApp nicht virtualisiert werden können 12
- Empfehlungen für die Installation von ThinApp 12
  - Verwenden eines neu aufgesetzten Computers 12
  - Verwenden des frühesten Betriebssystems, das für die Benutzer erforderlich ist 13
- Installation von ThinApp 13
- Suchen nach ThinApp-Installationsdateien 13

2 Kapselung von Anwendungen 15
- Phasen des Kapselungsvorgangs 15
- Vorbereiten der Kapselung von Anwendungen 15
- Kapseln von Anwendungen mit dem Setup Capture-Assistenten 16
  - Erstellen eines Systemabbilds vor der Anwendungsinstallation 16
  - Erneutes Überprüfen des Systems nach der Anwendungsinstallation 16
  - Definition von Einstiegspunkten als Verknüpfungen zur virtuellen Umgebung 17
  - Einstiegspunkte festlegen 17
  - Benutzergruppen einrichten 18
  - Definition von Isolationsmodi für das physische Dateisystem 18
  - Festlegen der Systemisolationsmodi 20
  - Speichern von Änderungen der Anwendung in der Sandbox 20
  - Anpassen des Speicherorts der Sandbox 21
  - Senden anonymer statistischer Daten an VMware 21
  - Anpassen der ThinApp-Projekteinstellungen 21
  - Definition von Paketeinstellungen 21
  - Anpassen der Paketeinstellungen 22
  - Öffnen von Projekt- und Parameterdateien 23
  - Erstellen von virtuellen Anwendungen 23
- Erweiterte Paketkonfiguration 24
  - Ändern der Einstellungen in der Package.ini-Datei 24
  - Ändern der Einstellungen in der ##Attributes.ini-Datei 25
- Richtlinien zum Erstellen von Microsoft Office 2007-Paketen 25
  - Anforderungen zum Erstellen von Microsoft Office 2007-Paketen 25
  - Kapselung von Microsoft Office 2007 26
  - Konfigurieren von Microsoft Office 2007 28

3 Bereitstellen von Anwendungen 29
- ThinApp-Bereitstellungsoptionen 29
  - Bereitstellung von ThinApp mit Bereitstellungstools 29
  - Bereitstellen von ThinApp in der VMware View-Umgebung 29
  - Bereitstellen von ThinApp auf Netzwerkfreigaben 30
  - Bereitstellen von ThinApp unter Verwendung von ausführbaren Dateien 30
- Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe 30
  - Auswirkung von Application Sync auf das Dienstprogramm thinreg.exe 31
  - Ausführen des Dienstprogramms thinreg.exe 31

Optionale thinreg.exe-Parameter 31
Erstellen einer MSI-Datenbank 33
Anpassen von MSI-Dateien mit Package.ini-Parametern 33
Ändern der Datei Package.ini zum Erstellen von MSI-Dateien 34
Steuern des Anwendungszugriffs mit Active Directory 35
Package.ini-Einträge für die Active Directory-Zugriffssteuerung 36
Starten und Anhalten von virtuellen Diensten 36
Verwenden von ThinApp-Paketen mit Netzwerkstreaming 37
So funktioniert Anwendungsstreaming mit ThinApp 37
Anforderungen und Empfehlungen für Streamingpakete 38
Streamen von ThinApp-Paketen über das Netzwerk 39
Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten 39
Ausführen von Einfügevorgängen 39
Zugriff auf Drucker 39
Zugriff auf Treiber 39
Zugriff auf die lokale Festplatte, den Wechseldatenträger und die Netzwerkfreigaben 40
Zugriff auf die Systemregistrierung 40
Zugriff auf Netzwerk und Sockets 40
Verwenden von gemeinsam genutztem Arbeitsspeicher und Named Pipes 40
Verwenden von COM-, DCOM- und Out-of-Process-COM-Komponenten 40
Starten von Diensten 41
Verwenden von Dateitypzuordnungen 41
Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext 41
Ansicht der Auswirkung des Isolationsmodus auf die Windows-Registrierung 41

4 Aktualisieren und Verknüpfen von Anwendungen 43
Anwendungs-Updates, die der Endbenutzer auslöst 43
Application Sync-Updates 43
Application Link-Updates 47
Anwendungs-Updates, die der Administrator auslöst 51
Erzwingen eines Updates von Application Sync auf Clientcomputern 52
Aktualisieren von Anwendungen mit Laufzeitänderungen 52
Automatische Anwendungs-Updates 53
Dynamische Updates ohne Administratorrechte 54
Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe 54
Dateisperren 54
Upgrade einer laufenden Anwendung 55
Sandbox-Überlegungen für aktualisierte Anwendungen 55
Aktualisieren der ThinApp-Version von Paketen 55
Relink-Beispiele 55

5 Konfigurieren von Paketparametern 57
Package.ini-Dateistruktur 58
Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden 58
Konfigurieren der ThinApp-Laufzeit 58
NetRelaunch 58
RuntimeEULA 59
VirtualComputerName 59
Wow64 60
QualityReportingEnabled 60
Konfigurieren des Isolationsmodus 60
DirectoryIsolationMode 60
RegistryIsolationMode 62
Konfigurieren von Datei- und Protokollzuordnungen 62
FileTypes 62

Protocols 63
Konfigurieren der Build-Ausgabe 63
ExcludePattern 63
Icon 64
OutDir 64
RetainAllIcons 64
Konfigurieren von Berechtigungen 65
AccessDeniedMsg 65
AddPageExecutePermission 65
PermittedGroups 66
UACRequestedPrivilegesLevel 67
UACRequestedPrivilegesUIAccess 67
Konfigurieren von Objekten und DLL-Dateien 67
ExternalCOMObjects 67
ExternalDLLs 68
ForcedVirtualLoadPaths 68
IsolatedMemoryObjects 69
IsolatedSynchronizationObjects 69
NotificationDLLs 70
NotificationDLLSignature 70
ObjectTypes 71
SandboxCOMObjects 71
VirtualizeExternalOutOfProcessCOM 71
Konfigurieren von Dateispeicher 72
CachePath 72
UpgradePath 73
VirtualDrives 73
Konfigurieren von Prozessen und Diensten 75
AllowExternalKernelModeServices 75
AllowExternalProcessModifications 75
AllowUnsupportedExternalChildProcesses 76
AutoShutdownServices 76
AutoStartServices 76
ChildProcessEnvironmentDefault 77
ChildProcessEnvironmentExceptions 77
Konfigurieren von Größen 77
BlockSize 77
CompressionType 78
MSICompressionType 79
OptimizeFor 79
Konfigurieren der Protokollierung 80
DisableTracing 80
LogPath 80
Konfigurieren von Versionen 81
CapturedUsingVersion 81
StripVersionInfo 81
Version.XXXX 81
Konfigurieren von Gebietsschemata 82
AnsiCodePage 82
LocaleIdentifier 82
LocaleName 82
Konfigurieren von einzelnen Anwendungen 83
CommandLine 83
Disabled 83
ReadOnlyData 84
ReserveExtraAddressSpace 84

Shortcut 84
Shortcuts 85
Source 85
WorkingDirectory 86
Konfigurieren von abhängigen Anwendungen mit Application Link 86
Pfadnamenformate für Application Link 86
RequiredAppLinks 87
OptionalAppLinks 88
Konfigurieren von Anwendungs-Updates mit Application Sync 88
AppSyncClearSandboxOnUpdate 89
AppSyncExpireMessage 89
AppSyncExpirePeriod 89
AppSyncURL 90
AppSyncUpdateFrequency 90
AppSyncUpdatedMessage 90
AppSyncWarningFrequency 90
AppSyncWarningMessage 91
AppSyncWarningPeriod 91
Konfigurieren von MSI-Dateien 91
MSIArpProductIcon 91
MSIDefaultInstallAllUsers 92
MSIFilename 92
MSIInstallDirectory 93
MSIManufacturer 93
MSIProductCode 93
MSIProductVersion 94
MSIRequireElevatedPrivileges 94
MSIUpgradeCode 95
MSIUseCabs 95
Konfigurieren von Sandbox-Speicher und Bestandsnamen 96
InventoryName 96
RemoveSandboxOnExit 97
SandboxName 97
SandboxNetworkDrives 97
SandboxPath 98
SandboxRemovableDisk 98

6 Suche nach der ThinApp-Sandbox 99
Suchreihenfolge für die Sandbox 99
Steuern des Sandbox-Speicherorts 101
Speichern der Sandbox im Netzwerk 101
Speichern der Sandbox auf einem portablen Gerät 101
Sandbox-Struktur 102
Änderungen an der Sandbox 102
Auflisten virtueller Registrierungsinhalte mit vregtool 102

7 Erstellen von ThinApp-Snapshots und -Projekten von der Befehlszeile 103
Methoden zur Verwendung des Dienstprogramms snapshot.exe 103
Erstellen von Snapshots des Computerstatus 103
Erstellen der Template Package.ini-Datei aus zwei Snapshot-Dateien 104
Erstellen des ThinApp-Projekts aus der Template Package.ini-Datei 104
Anzeigen von Inhalten einer Snapshot-Datei 105
Beispiele für snapshot.exe-Befehle 105
Erstellen eines Projekts ohne den Setup Capture-Assistenten 105
Anpassen der Snapshot.ini-Datei 106

8 ThinApp-Dateisystemformate und Makros 107
Virtuelle Dateisystemformate 107
ThinApp-Ordnermakros 107
Liste der ThinApp-Makros 108
Verarbeitung von %SystemRoot% in einer Terminaldienste-Umgebung 109

9 Erstellen von ThinApp-Skripts 111
Rückruffunktionen 111
Implementieren von Skripts in einer ThinApp-Umgebung 112
Beispiel: .bat 112
Beispiel: Timeout 113
Ändern der virtuellen Registrierung 113
Beispiel: .reg 113
Beispiel: Anhalten eines Dienstes 113
Beispiel: Kopieren einer Datei 114
Hinzufügen eines Wertes zur Systemregistrierung 114
API-Funktionen 115
AddForcedVirtualLoadPath 115
ExitProcess 116
ExpandPath 116
ExecuteExternalProcess 116
ExecuteVirtualProcess 117
GetBuildOption 117
GetFileVersionValue 118
GetCommandLine 119
GetCurrentProcessName 119
GetOSVersion 119
GetEnvironmentVariable 120
RemoveSandboxOnExit 120
SetEnvironmentVariable 121
SetfileSystemIsolation 121
SetRegistryIsolation 121
WaitForProcess 122

10 ThinApp – Überwachung und Problembehandlung 123
Bereitstellen von Informationen für den technischen Support 123
Protokoll-Monitor-Vorgänge 124
Problembehandlung mit Protokoll-Monitor 124
Ausführen von erweiterten Protokoll-Monitor-Vorgängen 125
Protokollformat 126
Problembehandlung bei bestimmten Anwendungen 131
Fehlerbehebung beim Setup der Registrierung für Microsoft Outlook 131
Anzeigen von Anhängen in Microsoft Outlook 131
Starten von Explorer.exe in der virtuellen Umgebung 132
Problembehandlung bei Versionskonflikten von Java Runtime Environment 133

Glossar 135

Index 139

Über dieses Handbuch

Das *ThinApp-Benutzerhandbuch* enthält Informationen zur Installation von ThinApp™ sowie die Kapselung, die Bereitstellung und das Upgrade von Anwendungen. Sie können auf dieses Handbuch zurückgreifen, wenn Sie Parameter individuell anpassen und Scripting vornehmen möchten.

Zielgruppe

Dieses Handbuch ist ein Leitfaden für alle, die ThinApp installieren und gekapselte Anwendungen bereitstellen. Typische Anwender sind Systemadministratoren, die für Vertrieb und Wartung von Firmensoftwarepaketen verantwortlich sind.

Feedback zu diesem Dokument

VMware freut sich über Kommentare und Anregungen, um seine Dokumentation weiter zu verbessern. Senden Sie Ihr Feedback bitte an docfeedback@vmware.com.

Technischer Support und Schulungsressourcen

In den folgenden Abschnitten werden die verfügbaren technischen Supportressourcen beschrieben. Um auf die aktuelle Version dieses Handbuchs und anderer Bücher zuzugreifen, besuchen Sie bitte http://www.vmware.com/support/pubs.

Online- und Telefonsupport

Im Online-Support können Sie technische Unterstützung anfordern, Ihre Produkt- und Vertragsdaten abrufen und Produkte registrieren. Weitere Informationen finden Sie unter http://www.vmware.com/support.

Kunden mit entsprechenden Supportverträgen erhalten über den Telefonsupport die schnellste Hilfe bei Problemen der Prioritätsstufe 1. Weitere Informationen finden Sie unter http://www.vmware.com/support/phone_support.

Supportangebote

VMware stellt ein umfangreiches Supportangebot bereit, um Ihre geschäftlichen Anforderungen zu erfüllen. Weitere Informationen finden Sie unter http://www.vmware.com/support/services.

VMware Professional Services

VMware-Schulungskurse umfassen umfangreiche Praxisübungen, Fallbeispiele und Kursmaterialien, die zur Verwendung als Referenz-Tools bei der praktischen Arbeit vorgesehen sind. Kurse werden vor Ort, im Schulungsraum und live über das Internet angeboten. Für Pilotprogramme vor Ort und die Implementierung von Best Practices verfügt VMware Consulting Services über Angebote, um Ihnen bei Bewertung, Planung, Aufbau und Verwaltung Ihrer virtuellen Umgebung zu helfen. Um Informationen über Schulungskurse, Zertifizierungsprogramme und Consultingdienste zu erhalten, besuchen Sie bitte http://www.vmware.com/services.

1 Installation von ThinApp

Mit der Installation von ThinApp können Sie Anwendungen isolieren, Anwendungsanpassungen vereinfachen, Anwendungen auf verschiedenen Betriebssystemen bereitstellen und Anwendungskonflikte eliminieren.

Dieser Abschnitt umfasst die folgenden Themen:

- „ThinApp-Installationsanforderungen“ auf Seite 11
- „Empfehlungen für die Installation von ThinApp“ auf Seite 12
- „Installation von ThinApp“ auf Seite 13
- „Suchen nach ThinApp-Installationsdateien“ auf Seite 13

ThinApp-Installationsanforderungen

Überprüfen Sie vor der Installation von ThinApp die Anforderungen an die Betriebssysteme und die gekapselten Anwendungen.

Betriebssysteme, Anwendungen und Systeme, die von ThinApp unterstützt werden

ThinApp unterstützt folgende Betriebssysteme, Anwendungen und Systeme:

- 32-Bit-Plattformen, einschließlich Windows NT, Windows 2000, Windows XP, Windows XPE, Windows 2003 Server, Windows Vista, Windows Server 2008, Windows 7
- 64-Bit-Plattformen, einschließlich Windows XP 64-Bit, Windows 2003 64-Bit, Windows Vista 64 -Bit, Windows Server 2008 64-Bit, Windows Server 2008 R2 64-Bit, Windows 7 64 -Bit
- Auf 32-Bit-Windows-Betriebssystemen ausgeführte 16-Bit-Anwendungen
- Auf 32-Bit- und 64-Bit-Windows-Betriebssystemen ausgeführte 32-Bit-Anwendungen
- Terminalserver und Citrix Xenapp

ThinApp unterstützt japanische Anwendungen, die auf japanischen Betriebssystemen gekapselt und ausgeführt werden.

Folgende Betriebssysteme und Anwendungen werden von ThinApp nicht unterstützt:

- 16-Bit- oder Nicht-x86-Plattformen wie Windows CE
- Auf 32-Bit- oder 64-Bit-Windows-Betriebssystemen ausgeführte 64-Bit-Anwendungen
- Auf 64-Bit-Windows-Betriebssystemen ausgeführte 16-Bit-Anwendungen

Anwendungen, die von ThinApp nicht virtualisiert werden können

ThinApp kann einige Anwendungen nicht in virtuelle Anwendungen konvertieren und blockiert möglicherweise bestimmte Anwendungsfunktionen.

Zur Bereitstellung der folgenden Anwendungstypen sind herkömmliche Installationstechniken erforderlich:

- Anwendungen, die die Installation von Gerätetreibern im Kernelmodus erfordern.

  ODBC-Treiber funktionieren, da sie Treiber im Benutzermodus sind.
- Antivirus und persönliche Firewalls
- Scannertreiber und Druckertreiber
- Einige VPN-Clients

Gerätetreiber

Anwendungen, für die Gerätetreiber erforderlich sind, funktionieren nach der Paketierung durch ThinApp nicht. Diese Gerätetreiber müssen in ihrem ursprünglichen Format auf dem Hostcomputer installiert werden. Da ThinApp keine virtualisierten Gerätetreiber unterstützt, können Sie ThinApp nicht verwenden, um Antivirus, VPN-Clients, persönliche Firewalls und Dienstprogramme für die Festplatte und zum Bereitstellen von Volumes zu virtualisieren.

Wird Adobe Acrobat gekapselt, können PDF-Dateien modifiziert und gespeichert werden, doch der PDF-Druckertreiber, der das Speichern von Dokumenten im PDF-Format ermöglicht, kann nicht verwendet werden.

Shell-Integration

Einige Anwendungen, die Shell-Integration bieten, erfahren möglicherweise Funktionsbeschränkungen, wenn sie in einem ThinApp-Paket vorhanden sind. So kann beispielsweise eine virtuelle Anwendung, die in Windows Explorer integriert ist, keine spezifischen Einträge zu den Windows Explorer-Kontextmenüs hinzufügen.

Über das Netzwerk verfügbare DCOM-Dienste

ThinApp isoliert COM- und DCOM-Dienste. Auf Anwendungen, die DCOM-Dienste installieren, kann auf dem lokalen Computer nur durch andere gekapselte Anwendungen zugegriffen werden, die in derselben ThinApp-Sandbox ausgeführt werden. ThinApp unterstützt virtuelles DCOM und COM auf demselben Computer; Netzwerk-DCOM wird jedoch nicht unterstützt.

Global Hook-DLLs (Dynamic Link Libraries)

Einige Anwendungen verwenden die API-Funktion `SetWindowsHookEx`, um sämtlichen Prozessen auf dem Hostcomputer eine DLL-Datei hinzuzufügen. Die DLL fängt Windows-Meldungen ab, um Tastatur- und Mauseingaben von anderen Anwendungen zu kapseln. ThinApp ignoriert Anforderungen von Anwendungen, die die Funktion `SetWindowsHookEx` für den Versuch einsetzen, globale Hook-DLLs zu installieren. ThinApp beschränkt möglicherweise die Funktionen der Anwendung.

Empfehlungen für die Installation von ThinApp

Berücksichtigen Sie bei der Installation von ThinApp die Empfehlungen und die bewährten Methoden.

Verwenden eines neu aufgesetzten Computers

VMware empfiehlt, für die Installation von ThinApp einen neu aufgesetzten Computer zu verwenden, da die Umgebung sich auf den Kapselungsvorgang der Anwendung auswirkt. Ein neu aufgesetzter Computer ist eine physische oder virtuelle Maschine, auf der nur ein Windows-Betriebssystem installiert ist. In einer Unternehmensumgebung mit einem Basis-Desktop-Bild, ist das Basis-Desktop-Bild ein neu aufgesetzter Computer. Möglicherweise wurden auf dem Desktop-Computer bereits einige Komponenten und Bibliotheken vorinstalliert.

Installationsassistenten von Anwendungen überspringen Dateien, die bereits auf dem Computer vorhanden sind. Überspringt der Installationsassistent Dateien, schließt das ThinApp-Paket sie nicht während der Kapselung der Anwendung ein. Die Anwendung kann möglicherweise nicht auf anderen Computern, auf denen die Dateien nicht vorhanden sind, ausgeführt werden. Ein neu aufgesetzter Computer ermöglicht während der Kapselung die schnelle Überprüfung des Dateisystems und der Registrierung des Computers.

Wenn Sie ThinApp installieren und eine Anwendung auf einem Computer kapseln, auf dem Microsoft .NET 2.0 bereits installiert ist, wird .NET 2.0 nicht in das ThinApp-Paket eingeschlossen. Die gekapselte Anwendung wird nur auf Computern ausgeführt, auf denen .NET 2.0 bereits installiert ist.

Verwenden virtueller Maschinen als neu aufgesetzter Computer

Die einfachste Weise, einen neu aufgesetzten Computer einzurichten, ist das Erstellen einer virtuellen Maschine. Sie können Windows auf der virtuellen Maschine installieren und einen Snapshot der gesamten virtuellen Maschine im sauberen Zustand erstellen. Nach der Kapselung einer Anwendung können Sie den Snapshot wiederherstellen und die virtuelle Maschine in den sauberen Zustand zurücksetzen, die bereit für die nächste Kapselung der Anwendung ist.

Zum Erstellen virtueller Maschinen können Sie VMware Workstation oder andere VMware-Produkte verwenden. Informationen über VMware-Produkte finden Sie auf der Website von VMware.

Verwenden des frühesten Betriebssystems, das für die Benutzer erforderlich ist

Installieren Sie ThinApp auf einem neu aufgesetzten Computer mit der frühesten Version des Betriebssystems, das unterstützt werden soll. In den meisten Fällen ist die früheste Plattform Windows 2000 oder Windows XP. Die meisten unter Windows XP gekapselten Pakete funktionieren unter Windows 2000. In einigen Fällen enthält Windows XP einige DLLs, die Windows 2000 fehlen. ThinApp schließt diese DLLs aus dem gekapselten Anwendungspaket aus, wenn die Anwendung diese DLLs typischerweise installiert.

Nachdem Sie ein ThinApp-Anwendungpaket erstellt haben, können Sie Dateien im Paket mit aktualisierten Versionen überschreiben und die Anwendung ohne den Kapselungsvorgang neu erstellen.

Installation von ThinApp

Verwenden der ausführbaren ThinApp-Datei zum Installieren von ThinApp.

So installieren Sie die ThinApp-Software:

1. Laden Sie ThinApp auf eine physische oder virtuelle neu aufgesetzte Machine mit Windows-System herunter.
2. Doppelklicken Sie auf die ausführbare ThinApp-Datei.
3. Klicken Sie im Dialogfeld **Patentlisten (Patent Lists)** auf **Weiter (Next)**.
4. Akzeptieren Sie die Lizenzvereinbarung, geben Sie die Seriennummer und einen Lizenzanzeigenamen ein. Dieser wird beim Öffnen von Anwendungen angezeigt, die ThinApp kapselt.
5. Klicken Sie auf **Installieren (Install)**.

   ThinApp wird installiert.

Suchen nach ThinApp-Installationsdateien

Bei der Installation von ThinApp wird unter `C:\Programme\VMware` das `VMware ThinApp`-Verzeichnis generiert. Sie können Dateien in diesem Verzeichnis zur Durchführung bestimmter Vorgänge – wie dem Starten des Dienstprogramms Protokoll-Monitor zur Ansicht der letzten Aktivitäten – verwenden.

Die folgenden wichtigen Dateien im `VMware ThinApp`-Verzeichnis haben Auswirkungen auf ThinApp-Vorgänge:

- **AppSync.exe** – Hält gekapselte Anwendungen auf dem Stand der neuesten verfügbaren Version.
- **logging.dll** – Generiert `.trace`-Dateien.

- **dll_dump.exe** – Listet alle gekapselten Anwendungen, die aktuell auf einem System ausgeführt werden.
- **log_monitor.exe** – Zeigt den Ausführungsverlauf und die Fehler einer Anwendung an.
- **relink.exe** – Aktualisiert vorhandene Pakete auf die neueste auf dem System installierte ThinApp-Version.
- **sbmerge.exe** – Verbindet in der Anwendungs-Sandbox eingetragene Laufzeitänderungen mit dem ThinApp-Projekt und aktualisiert die gekapselte Anwendung.
- **Setup Capture.exe** – Kapselt Anwendungen und konfiguriert sie mithilfe eines Assistenten.
- **snapshot.exe** – Vergleicht während der Kapselung der Anwendung die Umgebung vor und nach der Installation.

  ThinApp startet dieses Dienstprogramm während des Setup Capture-Prozesses.
- **snapshot.ini** – Speichert Einträge für die virtuelle Registrierung und das virtuelle Dateisystem, die ThinApp während des Kapselungsvorgangs einer Anwendung ignoriert.

  Die `snapshot.exe`-Datei verweist auf die `snapshot.ini`-Datei. Fortgeschrittene Benutzer können die `snapshot.ini`-Datei modifizieren, um sicherzustellen, dass ThinApp bestimmte Einträge bei der Erstellung eines Anwendungspakets nicht kapselt.
- **template.msi** – Erstellt die MSI-Dateien.

  Diese Vorlage kann angepasst werden, um sicherzustellen, dass die von ThinApp generierten `.msi`-Dateien den Bereitstellungsverfahren und -normen des Unternehmens gerecht werden. Zum Beispiel können Registrierungseinstellungen hinzufügt werden, die ThinApp als Bestandteil der Installation zu einem Clientcomputer hinzufügen soll.
- **thinreg.exe** – Registriert gekapselte Anwendungen auf einem Computer.

  Diese Registrierung beinhaltet das Einrichten von Verknüpfungen und des **Start**-Menüs und das Festlegen von Dateitypzuordnungen, die das Öffnen der Anwendungen ermöglichen.
- **tlink.exe** – Verbindet Schlüsselmodule während des Build-Prozesses der gekapselten Anwendung.
- **vftool.exe** – Kompiliert das virtuelle Dateisystem während des Build-Prozesses der gekapselten Anwendung.
- **vregtool.exe** – Kompiliert die virtuelle Registrierung während des Build-Prozesses der gekapselten Anwendung.

Kapselung von Anwendungen 2

Die Kapselung von Anwendungen dient dazu, eine Anwendung in eine virtuelle Umgebung zu paketieren.

Zur Kapselung von Anwendungen und zum Festlegen der anfänglichen Anwendungsparameter wird in erster Linie der Setup Capture-Assistent verwendet. Fortgeschrittene Benutzer, die Anwendungen von der Befehlszeile aus kapseln müssen, können anstelle des Setup Capture-Assistenten das Dienstprogramm `snapshot.exe` verwenden.

Dieser Abschnitt umfasst die folgenden Themen:

- „Phasen des Kapselungsvorgangs“ auf Seite 15
- „Vorbereiten der Kapselung von Anwendungen“ auf Seite 15
- „Kapseln von Anwendungen mit dem Setup Capture-Assistenten“ auf Seite 16
- „Erweiterte Paketkonfiguration“ auf Seite 24
- „Richtlinien zum Erstellen von Microsoft Office 2007-Paketen“ auf Seite 25

Phasen des Kapselungsvorgangs

Die Kapselung einer Anwendung umfasst Systemüberprüfungen, Anwendungskonfiguration, Paketkonfiguration und das Generieren der virtuellen Anwendung zur Verteilung.

Der Setup Capture-Assistent legt die anfänglichen Werte für die Anwendung fest. Die gesamten Parameter können außerhalb des Assistenten angepasst werden.

Vorbereiten der Kapselung von Anwendungen

Die Vorbereitung für die Kapselung von Anwendungen bedingt das Verständnis der Anforderungen und Abhängigkeiten der Anwendung.

Bei Zielanwendungen mit Abhängigkeiten zu anderen Anwendungen, Bibliotheken oder Frameworks können die Abhängigkeiten gekapselt oder das Dienstprogramm Application Link verwendet werden, um separate virtuelle Anwendungen während der Laufzeit zu verknüpfen. Informationen über das Dienstprogramm Application Link erhalten Sie unter „Application Link-Updates“ auf Seite 47.

Zielanwendungen, die ein Gebietsschema (beispielsweise ein spezielles Datumsformat) erfordern, können in einer Umgebung mit den erforderlichen Einstellungen des Gebietsschemas gekapselt werden. ThinApp führt virtuelle Anwendungen in Übereinstimmung mit den Gebiets- und Spracheinstellungen des Kapselungssystems aus. Die Einstellungen des Rechners, der die Anwendung ausführt, werden dabei nicht berücksichtigt. Obwohl die Standardeinstellung für das Gebietsschema durch Auskommentieren des Parameters `LocaleIdentifier` in der `Package.ini`-Datei und die erneute Erstellung der Anwendung angepasst werden kann, lassen sich Schwierigkeiten in der Kapselungsumgebung vermeiden. Informationen über den `LocaleIdentifier`-Parameter finden Sie unter „LocaleIdentifier“ auf Seite 82.

Kapseln von Anwendungen mit dem Setup Capture-Assistenten

Bei der Kapselung wird eine Anwendung paketiert und die anfänglichen Anwendungsparameter werden festgelegt. Wenn Sie eine virtuelle Maschine verwenden, empfiehlt es sich, vor dem Ausführen des Assistenten einen Snapshot zu erstellen. Ein Snapshot des ursprünglichen sauberen Zustands ermöglicht Ihnen das Zurücksetzen auf diesen Snapshot, wenn Sie eine weitere Anwendung kapseln möchten.

In den nachstehenden Erläuterungen dient Mozilla Firefox als Schlüsselbeispiel für die Kapselung von Anwendungen. Informationen über Microsoft Office 2007, die über den grundlegenden Kapselungsvorgang hinausgehen, finden Sie unter „Richtlinien zum Erstellen von Microsoft Office 2007-Paketen“ auf Seite 25.

Erstellen eines Systemabbilds vor der Anwendungsinstallation

Der Setup Capture-Assistent startet den Kapselungsvorgang mit einer Systemüberprüfung, um die Umgebung zu bewerten und um eine Baselineabbildung zu erstellen.

So erstellen Sie ein Systemabbild vor der Anwendungsinstallation

1 Laden Sie die zu kapselnden Anwendungen herunter.

Laden Sie beispielsweise `Firefox Setup 2.0.0.3.exe` herunter und kopieren Sie die Datei auf den neu aufgesetzten Computer, mit dem Sie arbeiten.

2 Schließen Sie alle Anwendungen, zum Beispiel Anti-Virenprogramme, die das Dateisystem während der Kapselung verändern könnten.

3 Wählen Sie auf dem Desktop **Start** > **Alle Programme** > **VMware** > **ThinApp Setup Capture**.

4 (Optional) Klicken Sie im Dialogfeld **Bereit zur Vorprüfung (Ready to Prescan)** auf **Erweiterte Speicherorte zur Überprüfung (Advanced scan locations)**, um die Laufwerke und Registrierungsstrukturen zum Überprüfen auszuwählen.

Es empfiehlt sich, einen anderen Speicherort als das Laufwerk `C:\` zu überprüfen, wenn Sie Anwendungen auf ein anderes Laufwerk installieren. In seltenen Fällen, wenn Sie wissen, dass das Installationsprogramm der Anwendung die Registrierung nicht verändert, empfiehlt es sich, die Registrierungsstruktur nicht zu überprüfen.

5 Klicken Sie auf **Vorprüfung (Prescan)**, um eine Baselineabbildung der Festplatte und der Registrierungsdateien zu erstellen.

Mit Windows XP dauert die Überprüfung ungefähr 10 Sekunden.

Erneutes Überprüfen des Systems nach der Anwendungsinstallation

Die zu virtualisierende Anwendung kann installiert werden, bevor der Setup Capture-Assistent das System erneut überprüft und Änderungen gegenüber dem ursprünglichen Systemabbild bewertet.

So installieren Sie die Anwendung und führen eine erneute Überprüfung des Systems durch

1 Wenn die Seite **Anwendung installieren (Install Application)** angezeigt wird, minimieren Sie den Setup Capture-Assistenten und installieren Sie die Anwendungen, die gekapselt werden sollen.

Doppelklicken Sie zum Beispiel auf `Firefox Setup 2.0.0.3.exe`, um Firefox zu installieren. Starten Sie das System neu, wenn die Anwendung nach der Installation neu gestartet werden muss. Bei diesem Vorgang wird der Setup Capture-Assistent erneut gestartet.

2 (Optional) Nehmen Sie erforderliche Konfigurationsänderungen vor, um die Richtlinien Ihres Unternehmens einzuhalten, beispielsweise die Festlegung spezieller Sicherheitseinstellungen oder einer bestimmten Startseite.

Wenn Sie zu diesem Zeitpunkt keine Konfigurationsänderungen vornehmen, müssen die einzelnen Benutzer Anpassungen vornehmen.

3 (Optional) Starten Sie die Anwendung und reagieren Sie auf alle Hinweise, bevor Sie mit dem Setup Capture-Assistenten fortfahren.

Wenn Sie zu diesem Zeitpunkt nicht auf Hinweise reagieren, müssen die einzelnen Benutzer der Anwendung dies beim ersten Starten tun.

4 Schließen Sie die Anwendung.

5 Maximieren Sie den Setup Capture-Assistenten, klicken Sie auf **Nachüberprüfung (Postscan)**, um mit einer weiteren Überprüfung des Computers fortzufahren, und klicken Sie dann auf **OK**, um den Nachüberprüfungsvorgang zu bestätigen.

ThinApp speichert die Unterschiede zwischen der ersten Baselineabbildung und dieser Abbildung in einem virtuellen Dateisystem und einer virtuellen Registrierung.

Definition von Einstiegspunkten als Verknüpfungen zur virtuellen Umgebung

Einstiegspunkte sind ausführbare Dateien, die als Verknüpfungen in die virtuelle Umgebung dienen und die virtuelle Anwendung starten. Welche Einstiegspunkte zur Auswahl stehen, hängt von den ausführbaren Dateien ab, die Ihre gekapselte Anwendung bei der Installation generiert.

Wenn Sie zum Beispiel Microsoft Office installieren, können Sie Einstiegspunkte wählen für Microsoft Word, Microsoft Excel und weitere Anwendungen, die während einer Microsoft Office-Installation installiert werden. Wenn Sie Firefox installieren, empfiehlt es sich, `Mozilla Firefox.exe` und `Mozilla Firefox (SafeMode).exe` zu wählen, falls Benutzer den Zugang im abgesicherten Modus benötigen.

Während des Build-Prozesses am Schluss des Setup Capture-Assistenten generiert ThinApp für jeden ausgewählten Einstiegspunkt eine ausführbare Datei. Wenn Sie die Anwendung als MSI-Datei bereitstellen oder das Dienstprogramm `thinreg.exe` verwenden, verweisen die auf den Endbenutzer-Desktops erstellten Verknüpfungen von Desktop und **Start**-Menü auf diese Einstiegspunkte.

Einstiegspunkte zur Problembehandlung

ThinApp bietet Einstiegspunkte für die Problembehandlung der Umgebung.

Für das Debugging einer Anwendung sind möglicherweise folgende Einstiegspunkte erforderlich:

- `cmd.exe` – Gibt eine Eingabeaufforderung in einem virtuellen Kontext aus, mit dem Sie das virtuelle Dateisystem anzeigen können.
- `regedit.exe` – Startet den Registrierungseditor in einem virtuellen Kontext, mit dem Sie die virtuelle Registrierung anzeigen können.
- `iexplore.exe` – Startet `iexplore.exe` in einem virtuellen Kontext, damit Sie virtualisierte ActiveX -Steuerungen testen können.

Einstiegspunkte starten native ausführbare Dateien in einem virtuellen Kontext. Einstiegspunkte erzeugen keine virtuellen Pakete von `cmd.exe`, `regedit.exe` oder `iexplore.exe`.

Wenn Sie nicht vorhersehen können, ob Debugging oder eine Problembehandlung der Umgebung erforderlich wird, können Sie stattdessen den `Disabled`-Parameter in der `Package.ini`-Datei verwenden, um diese Einstiegspunkte zu einem späteren Zeitpunkt zu aktivieren.

Einstiegspunkte festlegen

Sie können die ausführbaren Dateien für die Liste der Einstiegspunkte bestimmen. ThinApp installiert die ausführbaren Dateien während der Kapselung.

So legen Sie im Setup Capture-Assistenten Einstiegspunkte fest

1 Aktivieren Sie auf der Seite **Einstiegspunkte (Entry Points)** die Kontrollkästchen für Einstiegspunkte, auf die Benutzer Zugriff haben.

Der Assistent zeigt die ausführbaren Dateien an, auf die direkt über den Desktop oder über die Verknüpfungen des **Start**-Menüs zugegriffen werden kann.

2 (Optional) Um Ihre Umgebung zu debuggen, markieren Sie das Kontrollkästchen **Einstiegspunkte für Debugging anzeigen (Show entry points used for debugging)**, um die Optionen zur Fehlerbehebung `iexplore.exe`, `regedit.exe` und `cmd.exe` anzuzeigen.

Benutzergruppen einrichten

ThinApp kann mithilfe von Active Directory-Gruppen den Zugriff auf die virtuelle Anwendung autorisieren. Zum Beispiel können Sie den Zugriff auf eine Anwendung einschränken, um sicherzustellen, dass Benutzer diese nicht an unbefugte Benutzer übergeben.

Active Directory-Domänendienste definieren Sicherheitsgruppen und Verteilungsgruppen. ThinApp kann ausschließlich verschachtelte Sicherheitsgruppen unterstützen.

So legen Sie im Setup Capture-Assistenten Benutzergruppen fest

1 Beschränken Sie auf der Seite **Gruppen (Groups)** den Benutzerzugriff auf die Anwendung.

   a Wählen Sie **Nur die folgenden Active Directory-Gruppen (Only the following Active Directory groups)**.

   b Klicken Sie auf **Hinzufügen (Add)**, um die Objekt- und Speicherinformationen von Active Directory zu bestimmen.

Option	Beschreibung
Objekttypen (Object Types)	Gibt Objekte an.
Speicherorte (Locations)	Gibt einen Speicherort in der Gesamtstruktur an.
Überprüfen der Namen	Bestätigen der Objektnamen.
Erweitert (Advanced)	Sucht Benutzernamen in der Active Directory-Gesamtstruktur.
Häufig gestellte Fragen (Common Queries) (unter **Erweitert (Advanced)**)	Sucht nach Gruppen entsprechend den Namen, Beschreibungen, deaktivierten Konten, Passwörtern und Tagen seit der letzten Anmeldung.

2 (Optional) Ändern Sie die Mitteilung, die denjenigen Benutzern angezeigt wird, die ThinApp nicht autorisieren kann.

Definition von Isolationsmodi für das physische Dateisystem

Isolationsmodi bestimmen die Berechtigungsstufen für den Lese- und Schreibzugriff für das native Dateisystem außerhalb der virtuellen Umgebung. Abhängig von der Anwendung und den Anforderungen zum Schutz des physischen Systems vor Änderungen sollten Sie möglicherweise die Einstellungen der Isolationsmodi anpassen.

Die Auswahl der Isolationsmodi während des Kapselungsvorgangs bestimmt den Wert des `DirectoryIsolationMode`-Parameters in der `Package.ini`-Datei. Dieser Parameter steuert den Standard-Isolationsmodus für die von der virtuellen Anwendung erstellten Dateien, sofern Sie nicht einen anderen Isolationsmodus für ein einzelnes Verzeichnis in der `##Attributes.ini`-Datei bestimmen.

Die Auswahl eines Verzeichnis-Isolationsmodus wirkt sich auf die folgenden Bereiche nicht aus:

- ThinApp behandelt Schreibvorgänge auf Netzlaufwerke gemäß dem `SandboxNetworkDrives`-Parameter in der `Package.ini`-Datei. Dieser Parameter hat einen Standardwert, der Schreibvorgänge zum physischen Laufwerk lenkt. ThinApp behandelt Schreibvorgänge auf Wechseldatenträger gemäß dem `SandboxRemovableDisk`-Parameter in der `Package.ini`-Datei. Dieser Parameter hat einen Standardwert, der Schreibvorgänge zum physischen Laufwerk lenkt.
- Wenn Sie Dokumente auf den Desktop oder im Ordner `Eigene Dateien` speichern, speichert ThinApp die Dokumente auf dem physischen System. ThinApp legt den Isolationsmodus in den `##Attributes.ini`-Dateien, in `%Personal%` und `%Desktop%` auf `Zusammengeführt (Merged)` fest, auch dann, wenn Sie den Isolationsmodus „WriteCopy“ wählen.

Anwenden des Isolationsmodus „Zusammengeführt (Merged)“ für Änderungen außerhalb des Pakets

Mit dem Isolationsmodus „Zusammengeführt (Merged)“ kann eine Anwendung Elemente auf dem physischen Dateisystem außerhalb des virtuellen Pakets lesen und ändern. Einige Anwendungen sind vom Lesen der DLLs und Registrierungsdaten im lokalen Systemabbild abhängig.

Der Vorteil des Modus „Zusammengeführt (Merged)“ liegt darin, dass von Benutzern gespeicherte Dokumente im physischen System auf dem vom Benutzer erwarteten Speicherort anstatt in der Sandbox abgelegt werden. Der Nachteil ist, dass dieser Modus das Systemabbild überhäufen könnte. Ein Beispiel für die Störddaten können Markierungen für die Erstausführung von Shareware-Anwendungen auf zufälligen Computerspeicherorten als Teil des Lizenzierungsvorgangs sein.

Wenn Sie den Isolationsmodus „Zusammengeführt (Merged)“ auswählen, führt ThinApp folgende Vorgänge durch:

- Setzen des `DirectoryIsolationMode`-Parameters in der `Package.ini`-Datei auf `Zusammengeführt (Merged)`.
- Bestimmen von Ausnahmen, die den Isolationsmodus „WriteCopy“ für folgende Verzeichnisse und ihre Unterverzeichnisse anwenden:
  - `%AppData%`
  - `%Common AppData%`
  - `%Local AppData%`
  - `%Program Files Common%`
  - `%ProgramFilesDir%`
  - `%SystemRoot%`
  - `%SystemSystem%`

    Durch Erstellen einer Ausnahme für den Isolationsmodus des übergeordneten Verzeichnisses `%SystemSystem%` behält ThinApp den Isolationsmodus „Zusammengeführt (Merged)“ für das Unterverzeichnis `%SystemSytem%\spool` bei.
- Zwischen der Vor- und der Nachüberprüfung während des Kapselungsvorgangs weist ThinApp den Isolationsmodus „Voll (Full)“ jedem neuen Verzeichnis zu, das die Anwendung während der Installation erstellt. Dieser Vorgang ist unabhängig vom Isolationsmodus neuer Verzeichnisse, die von der aktuell ausgeführten virtuellen Anwendung erstellt werden.

Der Isolationsmodus „Zusammengeführt (Merged)“ im Setup Capture-Assistenten hat dieselbe Auswirkung wie der Isolationsmodus „Zusammengeführt (Merged)“ in der `Package.ini`-Datei, einschließlich der Ausnahmen im Verzeichnis, die den Isolationsmodus „WriteCopy“ bestimmen. Der Setup Capture-Assistent und der manuelle Kapselungsvorgang mit dem Dienstprogramm `snapshot.exe` konfigurieren mithilfe der `##Attributes.ini`-Dateien in den Verzeichnissen die Verzeichnisausnahmen für Sie.

Anwenden des Isolationsmodus „WriteCopy“, um Änderungen außerhalb des Pakets zu unterbinden

Mit dem Isolationsmodus „WriteCopy“ kann ThinApp Schreibvorgänge abfangen und in die Sandbox umleiten.

Sie können den Isolationsmodus „WriteCopy“ für ältere oder nicht vertrauenswürdige Anwendungen verwenden. Obwohl die Suche nach Benutzerdatendateien, die sich in der Sandbox statt im physischen System befinden, durch diesen Modus möglicherweise erschwert wird, ist dieser Modus für gesperrte Desktops nützlich, wenn Sie Benutzer daran hindern möchten, das lokale Dateisystem zu beeinflussen.

Wenn Sie im Setup Capture-Assistenten den Isolationsmodus „WriteCopy“ auswählen, führt ThinApp folgende Vorgänge durch:

- Setzen des `DirectoryIsolationMode`-Parameters in der `Package.ini`-Datei auf `Zusammengeführt (Merged)`.
- Bestimmen von Ausnahmen, die den Isolationsmodus „Zusammengeführt (Merged)“ auf folgende Verzeichnisse anwenden:
  - %Personal%
  - %Desktop%
  - %SystemSystem%\spool
- Zwischen der Vor- und der Nachüberprüfung während des Kapselungsvorgangs weist ThinApp den Isolationsmodus „Voll (Full)“ jedem neuen Verzeichnis zu, das die Anwendung während der Installation erstellt. Dieser Vorgang ist unabhängig vom Isolationsmodus neuer Verzeichnisse, die von der aktuell ausgeführten virtuellen Anwendung erstellt werden.

Der Isolationsmodus „WriteCopy“ im Setup Capture-Assistenten hat dieselbe Auswirkung wie der Isolationsmodus „WriteCopy“ in der `Package.ini`-Datei, einschließlich der Ausnahmen im Verzeichnis, die den Isolationsmodus „Zusammengeführt (Merged)“ bestimmen. Der Setup Capture-Assistent und das Dienstprogramm `snapshot.exe` konfigurieren mithilfe der `##Attributes.ini`-Dateien in den Verzeichnissen die Verzeichnisausnahmen für Sie.

Festlegen der Systemisolationsmodi

Der Kapselungsprozess legt die Berechtigungsstufen für den Lese- und Schreibzugriff für das physische Dateisystem fest, um zu bestimmen, welche Verzeichnisse von der virtuellen Anwendung gelesen oder geschrieben werden können.

Informationen zum Isolationsmodus „Voll (Full)“ und zur Registrierungsisolation, die nur außerhalb des Setup Capture-Assistenten zur Verfügung stehen, finden Sie unter „DirectoryIsolationMode“ auf Seite 60 und „RegistryIsolationMode“ auf Seite 62.

So legen Sie im Setup Capture-Assistenten die Systemisolationsmodi fest

Wählen Sie auf der Seite **Isolation** den Isolationsmodus für das physische Dateisystem.

Option	Beschreibung
Uneingeschränkter Schreibzugriff für Verzeichnisse mit Ausnahme von Systemverzeichnissen (Isolationsmodus „Zusammengeführt (Merged)“)	Erlaubt der Anwendung, Ressourcen auf dem lokalen Computer zu lesen und auf ihn zu schreiben.
Beschränkter Schreibzugriff (Isolationsmodus „WriteCopy“)	Erlaubt der Anwendung, Ressourcen auf dem lokalen Computer zu lesen und die meisten Änderungen auf die Sandbox zu beschränken. ThinApp kopiert Dateisystemänderungen in die Sandbox, um sicherzustellen, dass ThinApp nur Kopien anstatt der tatsächlichen physischen Dateien verändert.

Speichern von Änderungen der Anwendung in der Sandbox

Die Sandbox ist das Verzeichnis, in dem alle von der gekapselten Anwendung durchgeführten Änderungen gespeichert werden. Die Sandbox ist ein Speicher für Laufzeitänderungen und kein Cache. Sobald Sie die Anwendung das nächste Mal öffnen, werden diese Änderungen aus der Sandbox integriert.

Wenn Sie das Sandbox-Verzeichnis löschen, wird die Anwendung auf den gekapselten Status zurückgesetzt. Wenn eine Anwendung ein Problem hat und Sie sie wieder in den Originalzustand zurückversetzen möchten, sollten Sie die Sandbox löschen.

Anpassen des Speicherorts der Sandbox

Sie können die Sandbox auf einem lokalen Computer bereitstellen, auf einem USB-Gerät mit sich führen oder in einem Netzwerkpfad speichern.

Wenn Sie die Sandbox auf einem lokalen Computer bereitstellen, verwenden Sie als Speicherplatz für die Sandbox das Profil des Benutzers. Der Standardspeicherort der Sandbox für Firefox könnte `%AppData%\Thinstall\Mozilla Firefox 3.0` sein. Der typische `%AppData%`-Speicherort ist `C:\Dokumente und Einstellungen\<Benutzer_Name>\Anwendungsdaten`. Aufgrund des Schreibzugriffs ist das Benutzerprofil der Standardspeicherort.

Ein Netzwerkspeicherort ist nützlich, um die Sandbox zu sichern und ebenso für Benutzer, die sich an einem beliebigen Computer anmelden und ihre Anwendungseinstellungen beibehalten möchten. Verwenden Sie den absoluten Pfad zum Speicherort, beispielsweise `\\thinapp\sandbox\Firefox`. Sie können selbst dann einen Netzwerkspeicherort wählen, wenn eine Anwendung auf einem lokalen Computer installiert ist.

Ein Speicherort auf einem portablen Gerät ist nützlich, um die Sandbox-Daten auf dem Gerät zu speichern, auf dem sich die Anwendung befindet.

So passen Sie im Setup Capture-Assistenten die Sandbox-Speicherorte an

Wählen Sie auf der Seite **Sandbox** das Profil des Benutzers, das Anwendungsverzeichnis oder den Standardspeicherort für die Sandbox.

Senden anonymer statistischer Daten an VMware

Um ThinApps Support für Anwendungen zu verbessern, verwendet VMware den Kapselungsvorgang zur Bestätigung, ob anonyme Daten über bereitgestellte ThinApp-Pakete gesammelt werden dürfen. Diese Daten umfassen unter anderem die Startzeit der Anwendung, die gesamte Laufzeit und die Anzahl der Ausführungen der Anwendung.

So senden Sie anonyme Statistikdaten an VMware

Klicken Sie auf der Seite **Nutzungsstatistik (Usage Statistics)** auf die Optionsschaltfläche **Ja - Anonyme Nutzungsstatistik an VMware senden (Yes - Send anonymous usage statistics to VMware)**, um den Status zur Datensammlung zu bestätigen.

Anpassen der ThinApp-Projekteinstellungen

Ein Projekt umfasst die Daten, die durch den Kapselungsvorgang erstellt werden. Sie können die gekapselte Anwendung erst ausführen oder bereitstellen, nachdem Sie aus den Projektdateien ein Paket erstellt haben.

Beim Einrichten des Projekts muss der Bestandsname und der Projektspeicherort bestimmt werden. Der Bestandsname vereinfacht die interne Nachverfolgung der Anwendung und bestimmt den Standardnamen des Projektverzeichnisses.

So passen Sie im Setup Capture-Assistenten die Projekteinstellungen an

1 Ändern Sie auf der Seite **Projekteinstellungen (Project Settings)** den Bestandsnamen.

   Durch die Verwendung des Dienstprogrammes `thinreg.exe` oder die Bereitstellung der gekapselten Anwendung als MSI-Datei wird der Bestandsname im Dialogfeld **Software** von Windows angezeigt.

2 Ändern Sie das Verzeichnis, in dem Sie das ThinApp-Projekt speichern möchten.

   Wenn Sie das Standardverzeichnis behalten und Firefox 2.0.0.3 kapseln, wird als Pfad möglicherweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox (2.0.0.3)` angezeigt.

Definition von Paketeinstellungen

Ein Paket ist die ausführbare Datei oder die MSI-Datei mit ausführbaren Dateien, mit der Sie eine gekapselte Anwendung ausführen oder bereitstellen. Aus den Projektdateien erstellen Sie ein Paket.

Beim Einrichten des Pakets während der Kapselung müssen Angaben über die Hauptdatei der virtuellen Anwendung, die als primärer Datencontainer dient, die MSI-Generierung und die Komprimierung festgelegt werden.

Definition des primären Datencontainers

Der primäre Datencontainer ist die Hauptdatei der virtuellen Anwendung, in der die ThinApp-Laufzeit sowie das schreibgeschützte virtuelle Dateisystem und die virtuelle Registrierung enthalten sind. Die primäre Datencontainerdatei ist eine `.exe`-Datei oder eine `.dat`-Datei, die mit allen untergeordneten ausführbaren Dateien der Anwendung im selben `/bin`-Verzeichnis gespeichert ist. Einstiegspunkte beziehen sich auf die Informationen im primären Datencontainer.

Um nach der Kapselung einer Anwendung den primären Datencontainer zu identifizieren, prüfen Sie den `ReadOnlyData`-Parameter in der `Package.ini`-Datei.

Generieren von MSI-Paketen im Kapselungsvorgang

Sie können eine Anwendung kapseln und als MSI-Windows-Installationspaket bereitstellen. Die MSI-Installation speichert die Anwendungen im Verzeichnis `C:\Programme`.

Eine typische Firefox-Anwendung benötigt keine MSI-Installation. Andere Anwendungen, beispielsweise Microsoft Office, die sich in Anwendungs-Auslieferungstools integrieren, funktionieren problemlos als MSI-Paket. Für die MSI-Generierung müssen Sie MSI auf dem Zielgerät installieren, bevor Sie das Anwendungspaket verwenden können.

MSI-Pakete automatisieren den Registrierungsvorgang von Dateitypzuordnungen, die Registrierung von Verknüpfungen auf dem Desktop und im **Start**-Menü sowie die Anzeige von Systemsteuerungserweiterungen. Wenn Sie vorhaben, mit ThinApp ausführbare Dateien direkt auf jedem Computer bereitzustellen, können Sie mit dem Dienstprogramm `thinreg.exe` die gleiche Registrierung durchführen.

Die Komprimierung von Paketen im Kapselungsvorgang

Durch die Komprimierung eines Pakets während der Kapselung verringert sich die Größe eines ausführbaren Pakets; auf die MSI-Pakete hat dies jedoch keine Auswirkung.

Die Komprimierung kann den auf der Festplatte benötigten Speicherplatz um 50 Prozent verringern, sie verlangsamt jedoch die Anwendungsausführung, wenn ThinApp die ersten Blöcke dekomprimiert, die die Anwendung starten. VMware empfiehlt für Test-Builds keine Komprimierung, da die Komprimierung die Erstellzeit erhöht.

Anpassen der Paketeinstellungen

Der Kapselungsvorgang umfasst die ursprünglichen Einstellungen für den primären Datencontainer, die MSI-Pakete und die ausführbare Paket-Komprimierung.

So passen Sie im Setup Capture-Assistenten die Paketeinstellungen an

1 Wählen Sie auf der Seite **Paketeinstellungen (Package Settings)** aus der Liste, die sich aus den ausführbaren Dateieinstiegspunkten zusammensetzt, den primären Datencontainer aus.

   - Umfasst der primäre Datencontainer weniger als 200 MB, erstellt ThinApp eine `.exe`-Datei als primären Container. Für eine kleine Anwendung wie Firefox kann jede `.exe`-Datei als Hauptdatencontainer dienen.
   - Ist der primäre Datencontainer größer als 200 MB, erstellt ThinApp eine eigene `.dat`-Datei als primären Container, da Windows XP und Windows 2000 nicht über Verknüpfungssymbole für große `.exe`-Dateien verfügen. Das Problem lässt sich durch Generieren getrennter kleiner .`exe`-Dateien zusammen mit der `.dat`-Datei lösen.
   - Liegt die Größe der primären Containerdatei zwischen 200 MB und 1,5 GB, erstellt ThinApp die standardmäßige `.dat`-Datei, ausgenommen Sie wählen eine `.exe`-Datei, die die standardmäßige `.dat`-Datei außer Kraft setzt.

2 (Optional) Ignorieren Sie den angezeigten Warnhinweis, wenn Sie eine `.exe`-Datei zum Außerkraftsetzen der standardmäßigen `.dat`-Datei bei einer primären Containergröße von 200 MB bis 1,5 GB wählen.

Die Auswahl einer `.exe`-Datei ermöglicht es, alle Anwendungen ordnungsgemäß auszuführen; die korrekte Anzeige der Symbole könnte hierdurch jedoch verhindert werden.

3 (Optional) Wenn Sie keinen primären Datencontainer auswählen können, geben Sie einen primären Datencontainernamen ein, um eine `.dat`-Datei zu generieren.

Wenn die Verwendung des Dienstprogramms Application Sync geplant ist, um eine gekapselte Anwendung zu aktualisieren, verwendet ThinApp während des Vorgangs den primären Datencontainernamen. Sie müssen für mehrere Versionen der Anwendung denselben Namen verwenden. Möglicherweise können Sie denselben primären Datencontainernamen nicht aus der Liste wählen. Zum Beispiel haben Microsoft Office 2003 und Microsoft Office 2007 keine gemeinsamen Namen für Einstiegspunkte.

4 (Optional) Aktivieren Sie das Kontrollkästchen **MSI-Paket generieren (Generate MSI package)** und ändern Sie den MSI-Dateinamen.

5 (Optional) Um ein kleineres ausführbares Paket für Speicherorte wie zum Beispiel auf einem USB-Gerät zu erstellen, aktivieren Sie das Kontrollkästchen **Virtuelles Paket komprimieren (Compress virtual package)**.

6 Klicken Sie auf **Speichern (Save)**.

Öffnen von Projekt- und Parameterdateien

Der Kapselungsvorgang bietet Ihnen eine Gelegenheit, die Projektdateien zu überprüfen und vor dem Erstellen des ausführbaren Pakets oder des MSI-Pakets die Einstellungen zu aktualisieren.

Bei der Kapselung von Firefox 2.0.0.3 beispielsweise können Sie das Verzeichnis `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3` durchsuchen, um eine Einstellung (zum Beispiel eine Active Directory-Einstellung) in der `Package.ini`-Datei zu aktualisieren, die die während des Kapselungsvorgangs festgelegten Parameter enthält. Informationen über die Aktualisierung von Einstellungen erhalten Sie unter „Erweiterte Paketkonfiguration“ auf Seite 24.

Das Projekt umfasst Ordner, wie `%AppData%`, die Dateisystempfade darstellen und Speicherorte verändern können, wenn sie auf verschiedenen Betriebssystemen oder Computern ausgeführt werden. Die meisten Ordner enthalten `##Attributes.ini`-Dateien, die den Isolationsmodus auf der Ordnerebene bestimmen.

Erstellen von virtuellen Anwendungen

Sie können Projektdateien anpassen und die Anwendung für die Bereitstellung erstellen.

So erstellen Sie virtuelle Anwendungen im Setup Capture-Assistenten

1 (Optional) Überprüfen oder ändern Sie die Projektdateien auf der Seite **Bereit zum Erstellen (Ready to Build)**.

Option	Beschreibung
Bearbeiten der Package.ini-Datei	Modifizieren der Anwendungsparameter für das gesamte Paket.
Öffnen des Projektordners	Durchsuchen der ThinApp-Projektdateien in Windows Explorer.

2 (Optional) Um das Erstellen zu verhindern, aktivieren Sie das Kontrollkästchen **Erstellungsvorgang überspringen (Skip the build process)**.

Sie können das Paket zu einem späteren Zeitpunkt mithilfe der Datei `build.bat` im virtuellen Anwendungsordner erstellen. Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

3 Klicken Sie auf **Erstellen (Build)**, um ein ausführbares Paket oder eine MSI-Datei mit den während der Kapselung installierten Dateien zu erstellen.

4 (Optional) Deaktivieren Sie das Kontrollkästchen **Ordner mit den ausführbaren Projektdateien nach dem Klicken auf „Fertigstellen“ öffnen (Open folder containing project executables after clicking Finish)**, um die ausführbaren Dateien und MSI-Dateien zu einem späteren Zeitpunkt anzuzeigen.

5 Klicken Sie auf **Fertigstellen (Finish)**.

Nachdem Sie auf **Fertigstellen (Finish)** geklickt haben, können Sie das Paket jederzeit erneut erstellen, um Änderungen vorzunehmen.

Erweiterte Paketkonfiguration

Fortgeschrittene Anwender können Konfigurationsdateien wie die `Package.ini`-Dateien oder die `##Attributes.ini`-Dateien vor dem Erstellen des Pakets, während der Kapselung oder nach dem ursprünglichen Erstellen des Pakets modifizieren.

Ändern der Einstellungen in der Package.ini-Datei

Sie können die `Package.ini`-Datei modifizieren, um das gesamte Paket zu aktualisieren.

Die Datei ist im gekapselten Anwendungsordner gespeichert. Ein möglicher Firefox 2.0.0.3-Pfad lautet `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini`.

Die folgenden Parameter sind einige Beispiele für Einstellungen, die Sie ändern können:

- `DirectoryIsolationMode` – Setzt den Isolationsmodus auf `Merged (Zusammengeführt)`, `WriteCopy` oder `Full (Voll)`.
- `PermittedGroups` – Beschränkt die Verwendung eines Anwendungspakets auf eine spezifische Menge von Active Directory-Benutzern.
- `SandboxName` – Identifiziert die Sandbox.

  Sie können den Namen für inkrementelle Anwendungs-Updates beibehalten und für umfassende Updates ändern.
- `SandboxPfad` – Bestimmt den Speicherort der Sandbox.
- `SandboxNetworkDrives` – Gibt an, ob direkte Schreibvorgänge von der Netzwerkfreigabe in die Sandbox geleitet werden.
- `RequiredAppLinks` – Gibt eine Liste mit externen ThinApp-Paketen an, die während der Laufzeit in das aktuelle Paket importiert werden sollen.
- `OptionalAppLinks` – Gibt eine Liste mit externen ThinApp-Paketen an, die während der Laufzeit in das aktuelle Paket importiert werden sollen.

Allgemeine Informationen über sämtliche `Package.ini`-Parameter erhalten Sie in Kapitel 5, „Konfigurieren von Paketparametern“, auf Seite 57.

Ändern der Package.ini-Datei

Verwenden Sie einen Text-Editor, um die `Package.ini`-Datei zu ändern.

So ändern Sie die Package.ini-Datei

1 Öffnen Sie die `Package.ini`-Datei, die sich im Ordner der gekapselten Anwendung befindet.

Beispielsweise lautet ein möglicher Firefox 2.0.0.3-Pfad `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini`.

2 Aktivieren Sie den Parameter, den Sie bearbeiten möchten, indem Sie das Semikolon am Anfang der Zeile entfernen.

Aktivieren Sie beispielsweise für Firefox den `RemoveSandboxOnExit`-Parameter.

```
RemoveSandboxOnExit=1
```

3 Löschen oder ändern Sie den Wert des Parameters und speichern Sie die Datei.

4 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

Ändern der Einstellungen in der ##Attributes.ini-Datei

Die `##Attributes.ini`-Datei ist in den Ordnermakros des Projektordners vorhanden und wendet Konfigurationseinstellungen auf der Verzeichnisebene an. Die `Package.ini`-Datei wendet Einstellungen auf der gesamten Anwendungsebene an. Zum Überschreiben der `Package.ini`-Einstellungen auf der Verzeichnisebene können Sie die Parameter `DirectoryIsolationMode`, `CompressionType` und `ExcludePattern` in einer `##Attributes.ini`-Datei verwenden.

Sie können zum Beispiel den Isolationsmodus auf der Verzeichnisebene oder auf der Anwendungsebene festlegen, um zu bestimmen, welche Dateien und Registrierungsschlüssel durch die von Ihnen erstellte virtuelle Anwendung gelesen oder geschrieben werden können. Die detaillierte Einstellung in der `##Attributes.ini`-Datei setzt die allgemeine `Package.ini`-Einstellung außer Kraft. Die `Package.ini`-Einstellung legt den Isolationsmodus nur dann fest, wenn ThinApp keine `##Attributes.ini`-Informationen hat.

Die `##Attributes.ini`-Datei wird in den meisten Ordnern der gekapselten Anwendung angezeigt. Zum Beispiel könnte die `Attributes.ini`-Datei unter `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\%AppData%\##Attributes.ini` gespeichert sein.

Ändern der ##Attributes.ini-Datei

Verwenden Sie einen Text-Editor, um die `##Attributes.ini`-Datei zu ändern.

So ändern Sie die ##Attributes.ini-Datei

1 In der `##Attributes.ini`-Datei können Sie Parameter auskommentieren, aktualisieren oder löschen.

2 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Richtlinien zum Erstellen von Microsoft Office 2007-Paketen

Obwohl der Kapselungsvorgang für Microsoft Office 2007 von der jeweiligen Umgebung abhängen kann, können Sie auf grundlegende Richtlinien zurückgreifen.

Weitere Informationen über den Kapselungsvorgang und die Konfiguration für Microsoft Office und die Anforderungen seiner Umgebung finden Sie im ThinApp-Community Blog von VMware.

Anforderungen zum Erstellen von Microsoft Office 2007-Paketen

Die Anforderungen für den Kapselungsvorgang von Microsoft Office 2007 gehen über die Standardanforderungen zum Erstellen von ThinApp-Paketen hinaus.

Zum Erfüllen der folgenden Anforderungen ist Microsoft-Software und ein Drucker erforderlich:

- Eine neu aufgesetzte virtuelle Maschine mit einem unterstützten Windows-Betriebssystem.
- Eine lizenzierte Kopie von Microsoft Office 2007.

- Einen Volumenlizenzschlüssel für Microsoft Office 2007.

  Sie müssen auf jedem Computer, auf dem das Paket ausgeführt wird, einen Lizenzschlüssel aktivieren.

- Windows Installer 4.5.
- Microsoft .NET Framework 2.0.

  Sie müssen Microsoft .NET nach dem Vorprüfungsvorgang im Setup Capture-Assistenten und vor der Installation von Microsoft Office 2007 installieren.

- Erforderliche Drucker, beispielsweise ein Geschäftsdrucker, müssen vor dem Vorprüfungsvorgang im Setup Capture-Assistenten installiert werden.

Kapselung von Microsoft Office 2007

Die Hauptunterschiede zwischen der Kapselung von Microsoft Office 2007 und der Kapselung von allgemeinen Anwendungen bestehen in der Modifizierung von Microsoft Office und dem Blockieren untergeordneter Prozesse.

Der Kapselungsvorgang für Microsoft Office 2007 umfasst folgende Schritte:

1 „Anpassen der Installationsoptionen von Microsoft Office 2007“ auf Seite 26

2 „Deaktivieren von untergeordneten Prozessen in Microsoft Office 2007“ auf Seite 27

3 „Festlegen der Kapselungsoptionen für Microsoft Office 2007“ auf Seite 27

Dieser Vorgang setzt Kenntnisse in der Anwendung des Setup Capture-Assistenten voraus. Sie können den Vorgang in Übereinstimmung mit Ihrer Umgebung anpassen.

Anpassen der Installationsoptionen von Microsoft Office 2007

Der Start des Kapselungsvorgangs für Microsoft Office 2007 bedingt das Anpassen der Installation von Microsoft Office.

So passen Sie die Installation von Microsoft Office 2007 an

1 Kopieren Sie Microsoft .NET 2.0, Windows Installer 4.5 und die ThinApp-Installationsdateien auf den virtuellen Computer.

2 Kopieren Sie die Installationsdateien von Microsoft Office 2007 auf den virtuellen Computer oder laden Sie Microsoft Office 2007 ISO auf den virtuellen Computer.

3 Installieren Sie Windows Installer 4.5 und führen Sie einen Systemneustart durch.

4 Installieren Sie ThinApp.

5 (Optional) Installieren Sie erforderliche Drucker, beispielsweise einen Geschäftsdrucker.

6 Führen Sie den Setup Capture-Assistenten aus, bis der Vorprüfungsvorgang abgeschlossen wurde.

7 Minimieren Sie den Setup Capture-Assistenten, wenn die Seite **Anwendung installieren (Install Application)** des Assistenten angezeigt wird und installieren Sie Microsoft .NET 2.0.

8 Starten Sie den Installationsassistenten von Microsoft Office 2007.

  Die Installation bedingt die Eingabe des Lizenzschlüssels für Microsoft Office 2007 und die Annahme des Lizenzvertrags.

9 (Optional) Installieren Sie die Software auf der Registerkarte **Dateispeicherplatz** in einem festen Verzeichnis, beispielsweise `C:\Office`, anstatt in einem Standardverzeichnis, um so mögliche Zugriffsfehler der Online-Hilfe in Microsoft Office-Anwendungen zu verhindern.

  Die Zugriffsfehler beziehen sich nur auf Office-Anwendungen, die auf 64-Bit-Betriebssystemen bereitgestellt werden.

10 Klicken Sie im Dialogfeld **Auswahl der gewünschten Installation** auf die Schaltfläche **Anpassen**.

11 Passen Sie die Optionen auf der Registerkarte **Installationsoptionen** an, um die gewünschten Dateien zu kapseln und einen Druckerfehler von Microsoft Office auszuschließen.

   a Wählen Sie **Microsoft Office** > **Alle von Arbeitsplatz ausführen**.

   b Klicken Sie auf das Plussymbol (+) neben dem Menü **Office Tools**.

   c Klicken Sie auf das Plussymbol (+) neben dem Menü **Microsoft Office Document Imaging**.

   d Wählen Sie im Dropdownmenü **Microsoft Office Document Image Writer** die Option **Nicht verfügbar**.

   e Klicken Sie auf **Jetzt installieren**, um Microsoft Office 2007 zu installieren.

   f Schließen Sie den Installationsassistenten.

Deaktivieren von untergeordneten Prozessen in Microsoft Office 2007

Die Kapselung von Microsoft Office bedingt das Deaktivieren der untergeordneten Prozesse vor dem Nachüberprüfungsvorgang durch den Setup Capture-Assistenten.

Einige untergeordnete Prozesse verhindern das Schließen der Sandbox.

So deaktivieren Sie untergeordnete Prozesse in Microsoft Office 2007

1 Deaktivieren Sie das Ausführen der `ctfmon.exe`-Datei, um untergeordnete Prozesse daran zu hindern, das Schließen der Sandbox zu unterbinden.

   a Wählen Sie in der Windows-Systemsteuerung **Regions- und Sprachoptionen**.

   b Klicken Sie auf der Registerkarte **Sprachen** auf **Details**.

   c Markieren Sie auf der Registerkarte **Erweitert** das Kontrollkästchen **Erweiterte Textdienste deaktivieren**.

   d Klicken Sie auf dem Desktop auf **Start** > **Ausführen** und führen Sie den Befehl `Regsvr32.exe /u msimtf.dll` aus.

   e Klicken Sie auf dem Desktop auf **Start** > **Ausführen** und führen Sie den Befehl `Regsvr32.exe /u Msctf.dll` aus.

2 Deaktivieren Sie das Ausführen der `mdm.exe`-Datei, um Probleme mit untergeordneten Prozessen zu vermeiden, die Sie am Schließen der Sandbox hindern würden.

   a Wählen Sie im Internet Explorer **Extras** > **Internetoptionen**.

   b Markieren Sie auf der Registerkarte **Erweitert** das Kontrollkästchen **Skriptdebugging deaktivieren (Andere)** und das Kontrollkästchen **Skriptdebugging deaktivieren (Internet Explorer)**.

3 (Optional) Verwenden Sie die VMware Workstation zum Erstellen eines Snapshots der virtuellen Maschine.

   Diese Funktion erstellt ein Bild, das Ihnen das Zurücksetzen ermöglicht, wenn Sie Plug-Ins oder Updates hinzufügen.

Festlegen der Kapselungsoptionen für Microsoft Office 2007

Die letzte Phase des Kapselungsvorgangs für Microsoft Office 2007 bedingt den ThinApp-Nachüberprüfungsvorgang und die Optionen des Setup Capture-Assistenten zum Erstellen des Pakets.

Die folgenden Beispiele für Optionen können möglicherweise auf Microsoft Office 2007 angewendet werden:

- Der Speicherort der Einstiegspunkte lautet `%ProgramFilesDir%\Microsoft Office\Office12`.
- Der Name des primären Datencontainers lautet `Microsoft Office 2007.dat`.

- Der Isolationsmodus ist „Zusammengeführt (Merged)“.

  Der Isolationsmodus „WriteCopy“ ist der richtige, wenn Sie außerhalb der Sandbox keine Spuren von Dateien hinterlassen möchten.

- Die Bereitstellung von Microsoft Office 2007 verwendet ein MSI-Paket.

So legen Sie die Kapselungsoptionen für Microsoft Office 2007 fest

Maximieren Sie den Setup Capture-Assistenten und führen Sie den Kapselungsvorgang durch.

Konfigurieren von Microsoft Office 2007

Das Konfigurieren von Microsoft Office 2007 außerhalb des Kapselungsvorgangs bedingt das Löschen von Verzeichnissen und das Aktualisieren von Projektdateien.

Sie können Konfigurationsänderungen am Microsoft Office 2007-Paket vornehmen, wenn diese Änderungen sich für Ihre Umgebung eignen.

So konfigurieren Sie Microsoft Office 2007

- (Optional) Löschen Sie, um Platz zu sparen, die folgenden Verzeichnisse und Ordner, die Microsoft Office nicht benötigt:
  - `%COOKIES%`
  - `%HISTORY%`
  - `%INTERNET CACHE%`
  - `%PROFILE%`
  - `%COMMON APPDATA%\VMware`
  - `.msi`-Dateien und `.msp`-Dateien im `%SystemRoot%\Installer`
- (Optional) Wenn Sie an Ihrem Office 2007-Paket keine Anpassungen, die Benutzernamen oder Unternehmensnamen einschließen, vornehmen möchten, dann löschen Sie den Inhalt des `%APPDATA%`-Verzeichnisses mit Ausnahme der `##Attributes.ini`-Datei.

  Dieser Löschvorgang erzwingt eine saubere Konfiguration für den Benutzer.

- (Optional) Fügen Sie in der `HKEY_CURRENT_USER.txt`-Datei der Projektdateien an einer beliebigen Stelle der Datei den Eintrag `isolation_full HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Outlook\Sicherheit` hinzu, wenn er nicht bereits vorhanden ist.
- (Optional) Fügen Sie unterhalb des Eintrags `isolation_full HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Outlook\Sicherheit` folgende Einträge hinzu.

  ```
  Value=OutlookSecureTempFolder
  REG_SZ~%Profile%\Lokale Einstellungen\OutlookTemp#2300
  ```

- (Optional) Wenn Sie einen beliebigen Application Sync-Parameter und den `CompressionType`-Parameter aktivieren, deaktivieren Sie die Komprimierung in der `##Attributes.ini`-Datei des Verzeichnisses `%Program Files Common%\Microsoft Shared\OFFICE12\`.

  ```
  [Compression]
  CompressionType=None (Ohne)
  ```

  Ohne diese Änderung könnte sich ein Fehler auf die `%Program Files Common%\Microsoft Shared\OFFICE12\ODSERV.EXE`-Datei auswirken.

Bereitstellen von Anwendungen 3

Das Bereitstellen gekapselter Anwendungen erfordert die Verwendung von Bereitstellungstools, das Dienstprogramm `thinreg.exe`, MSI-Dateien sowie den Verzeichnisdienst Active Directory.

Dieser Abschnitt umfasst die folgenden Themen:

- „ThinApp-Bereitstellungsoptionen“ auf Seite 29
- „Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe“ auf Seite 30
- „Erstellen einer MSI-Datenbank“ auf Seite 33
- „Steuern des Anwendungszugriffs mit Active Directory“ auf Seite 35
- „Starten und Anhalten von virtuellen Diensten“ auf Seite 36
- „Verwenden von ThinApp-Paketen mit Netzwerkstreaming“ auf Seite 37
- „Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten“ auf Seite 39
- „Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext“ auf Seite 41

ThinApp-Bereitstellungsoptionen

Sie können gekapselte Anwendungen mit Bereitstellungstools, in einer VMware View™-Umgebung, auf einer Netzwerkfreigabe oder als grundlegende ausführbare Dateien bereitstellen.

Bereitstellung von ThinApp mit Bereitstellungstools

Mittelgroße und große Unternehmen verwenden häufig größere Bereitstellungstools, z. B. von Symantec, BMC und SMS. ThinApp funktioniert mit allen größeren Bereitstellungstools.

Wenn Sie eines dieser Tools verwenden, können Sie MSI-Dateien für die gekapselten Anwendungen erstellen und genau so vorgehen wie bei der Bereitstellung von nativen MSI-Dateien. Siehe die Bereitstellungsanleitungen der entsprechenden Händler. Informationen über MSI-Dateien erhalten Sie unter „Erstellen einer MSI-Datenbank“ auf Seite 33.

Bereitstellen von ThinApp in der VMware View-Umgebung

Sie können VMware View zur Verteilung von ThinApp-Paketen verwenden.

Der Workflow zur Bereitstellung von Paketen erfordert möglicherweise die folgenden Aufgaben:

- Erstellen von ausführbaren Dateien für die gekapselten Anwendungen.
- Speichern der ausführbaren Dateien auf einer Netzwerkfreigabe.

- Erstellen eines Anmeldeskripts, das Anwendungen, für die der Anwender berechtigt ist, abfragt und das Dienstprogramm `thinreg.exe` mit der Option ausführt, die die Anwendung auf dem lokalen Computer registriert. Anmeldeskripts sind insbesondere für nicht persistente Desktops nützlich. Siehe „Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe“ auf Seite 30.
- Steuerung des Benutzerzugriffs auf Dateifreigaben. IT-Administratoren möchten möglicherweise den Zugriff steuern, indem sie Netzwerkfreigaben nach Funktionen organisieren und die Berechtigung zum Zugriff auf Netzwerkfreigaben anhand dieser Funktionsbegrenzungen zuweisen.

Bereitstellen von ThinApp auf Netzwerkfreigaben

Kleine und mittelständische Unternehmen setzen häufig Netzwerkfreigaben ein. Sie können ausführbare Dateien für die gekapselte Anwendung erstellen und diese Dateien auf einer Netzwerkfreigabe speichern. Nun können Sie jedes Mal, wenn Sie eine neue Anwendung oder ein Update für ein vorhandenes Paket bereitstellen, Client-Benutzer benachrichtigen, damit sie das Dienstprogramm `thinreg.exe` mit einer entsprechenden Option ausführen.

IT-Administratoren können den Zugriff von Benutzern auf Dateifreigaben steuern, indem sie Netzwerkfreigaben nach Funktionen organisieren und die Berechtigung zum Zugriff auf Netzwerkfreigaben anhand dieser Funktionsbegrenzungen zuweisen.

Die Unterschiede zwischen den Optionen Netzwerkfreigabe und VMware View liegen darin, dass die Netzwerkfreigabe von einer Mischung aus physischen und virtuellen (persistenten) Desktops ausgeht und von den Benutzern erwartet wird, dass sie das Dienstprogramm `thinreg.exe` ausführen, anstatt sich auf Anmeldeskripts zu verlassen.

Bereitstellen von ThinApp unter Verwendung von ausführbaren Dateien

Ist die Festplattenbelegung eingeschränkt, können Sie eine grundlegende Bereitstellungsoption unter Verwendung von ausführbaren Dateien verwenden.

Sie können ausführbare Dateien für die gekapselten Anwendungen erstellen, sie von einem zentralen Repository kopieren und das Dienstprogramm `thinreg.exe` manuell ausführen, um die Dateitypenzuordnungen, Desktop-Verknüpfungen und das Anwendungspaket auf dem System zu registrieren.

Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe

Wenn Sie während des Kapselungsvorgangs ausführbare Dateien anstatt MSI-Dateien erstellen, müssen sie das Dienstprogramm `thinreg.exe` ausführen, um Dateien wie beispielsweise ein `.doc`-Dokument oder eine `.html`-Seite zu öffnen. Wenn Sie z. B. auf eine URL in einer E-Mail-Nachricht klicken, muss ThinApp so eingerichtet sein, dass Firefox geöffnet wird. Für MSI-Dateien müssen Sie das Dienstprogramm `thinreg.exe` nicht ausführen, weil MSI-Dateien das Dienstprogramm bereits während der Installation der Anwendung automatisch starten.

Das Dienstprogramm `thinreg.exe` erstellt das **Start**-Menü und die Desktop-Verknüpfungen, richtet Dateitypzuordnungen ein, fügt der Systemsteuerung Informationen zum Deinstallieren des Programms hinzu und entfernt die Registrierung von zuvor registrierten Paketen. Mit dem Dienstprogramm können Sie außerdem die Systemsteuerungserweiterungen für Anwendungen anzeigen, wie beispielsweise QuickTime, oder das Systemsteuerungs-Applet Mail für Microsoft Outlook 2007. Wenn Sie mit der rechten Maustaste auf eine `.doc`-Datei klicken, können Sie mit dem Dienstprogramm `thinreg.exe` dieselben Menüoptionen für eine `.doc`-Datei in einer nativen Umgebung anzeigen.

Wenn eine Anwendung SMTP- oder HTTP-Protokolle ausführt, wie beispielsweise einen E-Mail-Link auf einer Website, der in Microsoft Outlook 2007 geöffnet werden muss, startet das Dienstprogramm `thinreg.exe` verfügbare virtuelle Anwendungen, die diese Protokolle verarbeiten können. Falls keine virtuellen Anwendungen zur Verfügung stehen, startet das Dienstprogramm `thinreg.exe` native Anwendungen, die diese Protokolle verarbeiten können.

Der Standard-Speicherort des Dienstprogramms lautet `C:\Programme\VMware\VMware ThinApp`.

Auswirkung von Application Sync auf das Dienstprogramm thinreg.exe

Das Dienstprogramm Application Sync wirkt sich während des Update-Vorgangs auf das Dienstprogramm `thinreg.exe` aus.

Wenn Sie ausführbare Dateien hinzufügen, ändern oder entfernen, registriert das Dienstprogramm `thinreg.exe` die Dateitypzuordnungen, Verknüpfungen und Symbole erneut.

Wenn Sie Protokolle, MIME-Typen, Systemsteuerungs-Applets und Vorlagen, die keine ausführbaren Dateien sind, installieren, registriert das Dienstprogramm `thinreg.exe` diese Elemente erneut.

Ausführen des Dienstprogramms thinreg.exe

Dieses Beispiel für das Ausführen des Dienstprogramms `thinreg.exe` bietet einige Beispielbefehle.

Der Paketname in den `thinreg.exe`-Befehlen kann auf die folgende Art und Weise angezeigt werden:

- `C:\<Absoluter_Pfad_zu_.exe>`
- Relativer Pfad zur `.exe`-Datei
- `\\<Server>\<Freigabe>\<Pfad_zu_.exe>`

  Als Variation können Sie eine Platzhalterspezifikation verwenden, wie beispielsweise `*.exe`.

  Falls der Pfad oder Dateiname Leerstellen enthält, schließen Sie den Pfad in doppelte Anführungszeichen ein. Der folgende Befehl zeigt die Verwendung von doppelten Anführungszeichen.

  ```
  thinreg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office Word 2007.exe"
  ```

Informationen über `thinreg.exe`-Parameter erhalten Sie unter „Optionale thinreg.exe-Parameter“ auf Seite 31.

So führen Sie das Dienstprogramm thinreg.exe aus

1 Bestimmen Sie die ausführbaren Dateien, die ThinApp in der lokalen Umgebung registrieren muss.

2 Geben Sie in der Befehlszeile den Befehl `thinreg.exe` ein.

  `thinreg.exe [<optional_parameters>] [<package1.exe>][<package2.exe>][<packages_by_wildcard>]`

  Falls der Servername DEPLOYSERVER und die Freigabe `ThinApps` lautet, verwenden Sie das folgende Beispiel, um Microsoft Word für den angemeldeten Benutzer zu registrieren.

  `ThinReg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office 2007 Word.exe"`

  Verwenden Sie das folgende Beispiel, um alle Microsoft Office-Anwendungen in dem angegebenen Verzeichnis für den angemeldeten Benutzer zu registrieren.

  `ThinReg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office *.exe"`

Optionale thinreg.exe-Parameter

Das Dienstprogramm `thinreg.exe` überwacht die Einstellung `PermittedGroups` in der Datei `Package.ini` und registriert bzw. entfernt die Registrierung nach Bedarf. Wenn das Dienstprogramm `thinreg.exe` ein Paket für den aktuellen Benutzer registriert, erstellt das Dienstprogramm nur die Verknüpfungen und die Dateitypzuordnungen, für die der aktuelle Benutzer in der Einstellung `PermittedGroups` berechtigt ist. Falls diese Einstellung nicht vorhanden ist, ist der aktuelle Benutzer für alle ausführbaren Dateien berechtigt.

Wenn das Dienstprogramm `thinreg.exe` ein Paket für alle Benutzer mit dem Parameter `/allusers` registriert, erstellt ThinApp alle Verknüpfungen und Dateitypzuordnungen, unabhängig von der Einstellung in `PermittedGroups`. Wenn Sie auf ein Verknüpfungssymbol klicken, für das Sie keine Berechtigung besitzen, können Sie die Anwendung nicht ausführen.

Enthält der Paketname, den Sie registrieren bzw. für den Sie die Registrierung entfernen möchten, Leerstellen, so müssen Sie den Paketnamen in doppelte Anführungszeichen einschließen.

Informationen über die Einstellung `PermittedGroups` sowie Unterstützung für Active Directory-Gruppen erhalten Sie unter „PermittedGroups“ auf Seite 66.

Tabelle 3-1 listet optionale Parameter für das Dienstprogramm `thinreg.exe` auf. Für alle Befehle, die den Parameter `/a` verwenden, sind Administratorrechte erforderlich.

Tabelle 3-1. Optionale thinreg.exe-Parameter

Parameter	Zweck	Beispiel für die Verwendung
`/a`, `/allusers`	Registriert ein Paket für alle Benutzer. Falls ein nicht autorisierter Benutzer versucht, die Anwendung auszuführen, wird dieser durch eine Meldung informiert, dass er die Anwendung nicht ausführen kann.	`thinreg.exe /a "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
`/q`, `/quiet`	Verhindert die Anzeige einer Fehlermeldung für einen nicht erkannten Befehlszeilenparameter.	`thinreg.exe /q <Unbekannte_Option>`

`/u`, `/unregister`, `/uninstall`	Entfernt die Registrierung für ein Paket.
	Mit diesem Befehl wird die Software aus dem Systemsteuerungs-Applet „Software“ entfernt.
		Registrierung für Microsoft Word für den aktuellen Benutzer entfernen.
		`thinreg.exe /u "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
		Registrierung für alle Microsoft Office-Anwendungen für den aktuellen Benutzer entfernen und den Eintrag unter „Software“ entfernen.
		`thinreg.exe /u "\\server\share\Microsoft Office *.exe"`
		Falls ein Benutzer das Paket mit dem Parameter `/a` registriert, müssen Sie den Parameter `/a` auch beim Aufheben der Registrierung des Pakets verwenden.
		`thinreg.exe /u /a *.exe`

`/r`, `/reregister`	Registriert ein Paket erneut.
	Unter normalen Umständen kann das Dienstprogramm `thinreg.exe` erkennen, ob ein Paket bereits registriert ist, und es überspringen. Mit der Option `/r` wird das Dienstprogramm `thinreg.exe` dazu gezwungen, das Paket erneut zu registrieren.
		`thinreg.exe /r "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
		Falls ein Benutzer das Paket mit dem Parameter `/a` registriert, müssen Sie den Parameter `/a` auch bei der Neuregistrierung des Pakets verwenden.
		`thinreg.exe /r /a *.exe`

`/k`, `/keepunauthorized`, `/keep`	Verhindert das Entfernen der Registrierungsinformationen, auch wenn Sie keine Berechtigung mehr zum Zugriff auf ein Anwendungspaket besitzen. Ohne diese Option entfernt das Dienstprogramm `thinreg.exe` die Registrierungsinformationen für dieses Paket, wenn es erkennt, dass Sie keine Berechtigung mehr zum Zugriff auf das Paket besitzen. ThinApp speichert die Autorisierungsinformationen im Parameter `PermittedGroups` der `Package.ini`-Datei.	`thinreg.exe /k "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`

Tabelle 3-1. Optionale thinreg.exe-Parameter (Fortsetzung)

Parameter	Zweck	Beispiel für die Verwendung
`/noarp`	Verhindert das Erstellen eines Eintrags in dem Systemsteuerungs-Applet „Software“.	`thinreg.exe /q /noarp "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
`/norelaunch`	Startet das Dienstprogramm `thinreg.exe` auf Microsoft Vista ohne erhöhte Rechte. Standardbenutzer können das Dienstprogramm ohne ein Popup-Fenster für die Benutzerkontensteuerung (User Account Control, UAC) starten. Wenn die Notwendigkeit für weitere Rechte mit dem Dienstprogramm `thinreg.exe` erkannt wird, wie beispielsweise die für den Parameter `/allusers` erforderlichen Berechtigungen, wird das Dienstprogramm erneut gestartet und generiert ein Popup-Fenster für die UAC. Die Option `/norelaunch` blockiert diesen Neustartvorgang und führt zu einer fehlgeschlagenen Registrierung.	`thinreg.exe /q /norelaunch "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`

Erstellen einer MSI-Datenbank

Wenn Sie während des Kapselungsvorgangs keine MSI-Dateien erstellen, können Sie diese Dateien nach dem Erstellen einer Anwendung erstellen. Eine MSI-Datenbank ist nützlich für die Verteilung von gekapselten Anwendungen über herkömmliche Desktop-Managementsysteme an Remotestandorte und zur automatischen Erstellung von Verknüpfungen und Dateitypzuordnungen. Grundlegende Active Directory-Gruppenrichtlinien bieten Möglichkeiten zur Verteilung und zum Starten von MSI-Paketen.

ThinApp erstellt eine MSI-Datenbank, die gekapselte ausführbare Dateien, Installationslogik und das Dienstprogramm`thinreg.exe` enthält.

Anpassen von MSI-Dateien mit Package.ini-Parametern

Sie können das Verhalten der MSI-Dateien durch Modifizieren der `Package.ini`-Parameter und das erneute Erstellen des Anwendungspakets anpassen.

Die folgenden Parameter können sich auf die MSI-Konfiguration auswirken:

- Der Parameter `MSIInstallDirectory` richtet das Installationsverzeichnis für das Paket ein.

  Schließen Sie beispielsweise `MSIInstallDirectory=C:\Programme\` in die `Package.ini`-Datei ein.

- Der Parameter `MSIDefaultInstallAllUsers` richtet die Installation des Pakets für individuelle Benutzer ein. ThinApp installiert das Paket im Benutzerverzeichnis `%AppData%`.

  Schließen Sie beispielsweise `MSIDefaultInstallAllUsers=0` in die `Package.ini`-Datei ein.

  Weitere Informationen über diesen Parameter erhalten Sie unter „Festlegen einer Datenbankinstallation für einzelne Benutzer und Rechner“ auf Seite 34.

- Der Parameter `MSIFileName` vergibt einen Namen für das Paket.

  Schließen Sie beispielsweise `MSIFilename=Firefox30.msi` in die `Package.ini`-Datei ein.

- Der Parameter `MSIRequireElevatedPrivileges` gibt an, ob das Installationsprogramm erhöhte Rechte für die Bereitstellung auf Microsoft Vista benötigt. Installationen für individuelle Benutzer erfordern normalerweise keine erhöhten Rechte, aber Installationen, die für den ganzen Rechner erfolgen, benötigen solche Rechte.

  Schließen Sie beispielsweise `MSIRequireElevatedPrivileges=1` in die `Package.ini`-Datei ein.

- Der Parameter `MSIProductCode` vereinfacht die Installation einer neuen Version der Anwendung. Eine MSI-Datenbank enthält einen Produktcode und einen Upgrade-Code. Wenn Sie ein Paket aktualisieren, behalten Sie den ursprünglichen Wert des Parameters `MSIUpgradeCode` bei.

  Ist der Parameterwert der neuen Version derselbe wie der Wert der alten Version, so werden Sie von der Installation dazu aufgefordert, die alte Version zu entfernen. Wenn die Werte für den Parameter unterschiedlich sind, wird die alte Version deinstalliert und die neue Version installiert.

  VMware empfiehlt, dass Sie keinen Wert für den `MSIProductCode` festlegen und stattdessen ThinApp erlauben, für jede Version einen eigenen Produktcode zu generieren.

Unabhängig von den Parameterwerten, die beim Erstellungszeitpunkt festgelegt wurden, können Sie diese Einstellungen zum Zeitpunkt der Bereitstellung überschreiben. Siehe „Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner“ auf Seite 34. Weitere Informationen über MSI-Parameter erhalten Sie unter „Konfigurieren von MSI-Dateien“ auf Seite 91.

Ändern der Datei Package.ini zum Erstellen von MSI-Dateien

Zum Generieren von MSI-Dateien müssen Sie einen Wert für den Parameter `MSIFilename` hinzufügen.

Weitere Informationen über MSI-Parameter erhalten Sie unter „Anpassen von MSI-Dateien mit Package.ini-Parametern“ auf Seite 33 und „Konfigurieren von MSI-Dateien“ auf Seite 91.

So ändern Sie die MSI-Parameter

1 Geben Sie in der `Package.ini`-Datei den MSI-Dateinamen ein.

```
MSIFilename=<Dateiname>.msi
```

Der Dateiname für Firefox könnte beispielsweise `Mozilla Firefox 2.0.0.3.msi` lauten.

2 (Optional) Aktualisieren Sie die weiteren MSI-Parameter.

3 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Festlegen einer Datenbankinstallation für einzelne Benutzer und Rechner

Sie können die Installation der MSI-Datenbank für Benutzer und Rechner anpassen.

ThinApp installiert die MSI-Datenbank über alle Rechner hinweg. Sie können die Standardinstallation mit den folgenden Parameterwerten ändern:

- Um eine Datenbankinstallation für einzelne Benutzer zu erstellen, verwenden Sie den Wert 0 für den Parameter `MSIDefaultInstallAllUsers` in der `Package.ini`-Datei. Dieser Wert erstellt `msiexec`-Parameter für jeden Benutzer.
- Um Administratoren das Erstellen einer Datenbankinstallation für sämtliche Benutzer auf einem Rechner zu ermöglichen, oder um einem einzelnen Benutzer ohne Administratorenrechte das Erstellen einer Installation nur für sich selbst zu ermöglichen, verwenden Sie den Wert 2 für den Parameter `MSIDefaultInstallAllUsers`. Administratoren gehören zur Active Directory-Gruppe der Administratoren.

Weitere Informationen über den Parameter `MSIDefaultInstallAllUsers` erhalten Sie unter „MSIDefaultInstallAllUsers“ auf Seite 92.

Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner

Unabhängig von den Parameterwerten, die beim Erstellungszeitpunkt festgelegt wurden, können Sie die MSI-Einstellungen zum Zeitpunkt der Bereitstellung überschreiben.

Wenn Sie beispielsweise die Datenbank mit einem Wert 1 für den Parameter `MSIDefaultInstallAllUsers` erstellt haben, können Sie trotzdem individuelle Benutzerbereitstellungen für Firefox 3.0 erzwingen, indem Sie den Befehl `msiexec /i Firefox30.msi ALLUSERS=""` eingeben.

Wenn Sie das Argument `ALLUSERS=""` für den Befehl `msiexec` verwenden, extrahiert ThinApp die gekapselten ausführbaren Dateien in das Benutzerverzeichnis `%AppData%`.

So erzwingen Sie MSI-Bereitstellungen für einzelne Benutzer

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi ALLUSERS=""` ein.

So erzwingen Sie MSI-Bereitstellungen für alle Benutzer eines Rechners

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi ALLUSERS=1` ein.

Überschreiben des MSI-Installationsverzeichnisses

Sie können den Befehl `msiexec` verwenden, um das MSI-Standard-Installationsverzeichnis zu überschreiben.

Wenn ThinApp eine MSI-Bereitstellung für einen einzelnen Rechner durchführt, ist das Standard-Installationsverzeichnis das lokalisierte Äquivalent von `%ProgramFilesDir%\<Bestand_Name> (VMware ThinApp)`. Wenn Sie ein Firefox-Paket für jeden Rechner installieren, befindet sich das Paket auf `%ProgramFilesDir%\Mozilla Firefox (VMware ThinApp)`.

Wenn ThinApp eine MSI-Bereitstellung für einzelne Benutzer ausführt, lautet das Standard-Installationsverzeichnis `%AppData%\<Betand_Name> (VMware ThinApp)`.

In beiden Fällen können Sie das Installationsverzeichnis überschreiben, indem Sie die Eigenschaft INSTALLDIR zum Befehl `msiexec` übergeben.

So überschreiben Sie das MSI-Installationsverzeichnis

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi INSTALLDIR=C:\<Mein_Verzeichnis>\<Mein_Packet>` ein.

Bereitstellen von MSI-Dateien auf Microsoft Vista

Wenn Sie MSI-Dateien auf Vista bereitstellen, müssen Sie angeben, ob ein Installationsprogramm erhöhte Rechte erhalten muss. Typische Installationen für einzelne Benutzer erfordern keine erhöhten Rechte, aber Installationen, die für den gesamten Rechner erfolgen, benötigen solche Rechte.

ThinApp stellt den Parameter `MSIRequireElevatedPrivileges` in der `Package.ini`-Datei zur Verfügung, der die Anforderung erhöhter Rechte festlegt, wenn der Wert auf 1 gesetzt ist. Wenn Sie den Wert 1 für diesen Parameter festlegen oder eine einzelne Benutzerinstallation über die Befehlszeile erzwingen, können hierdurch UAC-Eingabeaufforderungen generiert werden. Wenn Sie den Wert 0 für diesen Parameter festlegen, werden die UAC-Eingabeaufforderungen verhindert, aber die Bereitstellung für rechnerweite Installationen schlägt fehl.

Steuern des Anwendungszugriffs mit Active Directory

Sie können den Zugriff auf Anwendungen mithilfe von Active Directory-Gruppen steuern.

Wenn Sie ein Paket erstellen, konvertiert ThinApp die Active Directory-Gruppennamen in SID-Werte. Eine SID (Security Identifier) ist ein kleiner binärer Wert, der ein Objekt eindeutig identifiziert. SID-Werte sind für einige Gruppen, wie die Administratorengruppe, nicht eindeutig. Da ThinApp die SID-Werte für die zukünftige Validierung in Paketen speichert, muss bei der Verwendung von Active Directory Folgendes in Betracht gezogen werden:

- Sie müssen während des Erstellungsvorgangs mit Ihrer Active Directory-Domäne verbunden sein und die Gruppen, die Sie festlegen, müssen vorhanden sein. ThinApp sucht während der Erstellung nach den SID-Werten.
- Wenn Sie eine Gruppe löschen und neu erstellen, könnte sich die SID ändern. Erstellen Sie in diesem Fall das Paket neu, um eine Authentifizierung der neuen Gruppe zu ermöglichen.

- Wenn Benutzer offline sind, kann ThinApp sie mithilfe von zwischengespeicherten Anmeldeinformationen authentifizieren. Wenn die Benutzer sich an ihren Rechnern anmelden können, funktioniert die Authentifizierung weiterhin. Verwenden Sie eine Gruppenrichtlinie, um den Zeitraum festzulegen, für den die zwischengespeicherten Anmeldeinformationen gültig sind.
- Zwischengespeicherte Anmeldeinformationen werden auf Clients möglicherweise erst beim nächsten Active Directory-Aktualisierungszyklus aktualisiert. Sie können eine Gruppenrichtlinie für einen Client erzwingen, indem Sie den Befehl `gpupdate` verwenden. Dieser Befehl aktualisiert die lokale Gruppenrichtlinie, die Gruppenrichtlinie sowie die Sicherheitseinstellungen, die in Active Directory gespeichert sind. Möglicherweise müssen Sie sich abmelden, bevor die Active Directory-Anmeldeinformationen erneut zwischengespeichert werden..
- Bestimmte Gruppen wie die Gruppen „Administratoren (Administrators)“ und „Alle (Everyone)“ haben in jeder Active Directory-Domäne und -Arbeitsgruppe dieselbe SID. Andere Gruppen, die Sie erstellen, haben eine domänenspezifische SID. Benutzer können keine eigene lokale Gruppe desselben Namens erstellen, um die Authentifizierung zu umgehen.
- Active Directory-Domänendienste definieren Sicherheitsgruppen und Verteilungsgruppen. Wenn Sie verschachtelte Gruppen verwenden, kann ThinApp ausschließlich verschachtelte Sicherheitsgruppen unterstützen.

Package.ini-Einträge für die Active Directory-Zugriffssteuerung

ThinApp stellt den Parameter `PermittedGroups` in der `Package.ini`-Datei zur Verfügung, um den Zugriff auf Active Directory zu steuern.

Wenn Sie eine gekapselte Anwendung starten, überprüft der Parameter `PermittedGroups`, ob der Benutzer ein Mitglied einer spezifizierten Active Directory-Gruppe ist. Falls der Benutzer kein Mitglied der Active Directory-Gruppe ist, startet ThinApp die Anwendung nicht. Informationen über die Beschränkung von Paketen auf Active Directory-Gruppen erhalten Sie unter „PermittedGroups“ auf Seite 66.

Im folgenden `Package.ini`-Eintrag erben App1 und App2 die Werte der `PermittedGroups` .

```
[BuildOptions]
PermittedGroups=Administratoren;Office-Benutzer
[App1.exe]
                ...
                ..
[App2.exe]
                ...
                ...
```

Im folgenden Eintrag können ausschließlich die zur Gruppe `App1users` gehörenden Benutzer die Datei `App1.exe` verwenden; Mitglieder der Gruppe `Alle (Everyone)` können die Datei `App2.exe` verwenden. Die Standardmeldung für abgelehnte Benutzer ändert sich für App1.

```
[BuildOptions]
PermittedGroups=Alle
[App1.exe]
PermittedGroups=App1-Benutzer
AccessDeniedMsg=Sie können diese Anwendung leider nicht ausführen
..
[App2.exe]
...
...
```

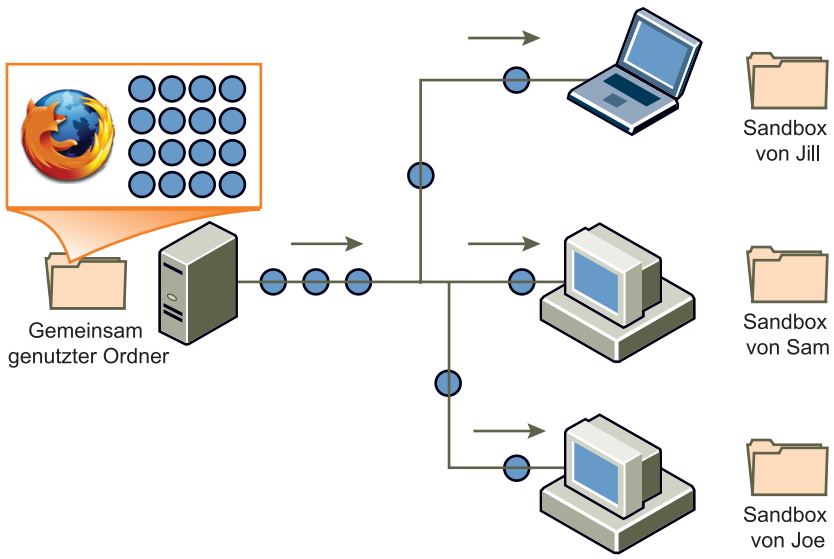
Starten und Anhalten von virtuellen Diensten

Wenn Sie ein Paket kapseln und bereitstellen, das einen Windows-Dienst nutzt, beispielsweise den SQL Server-Dienst, kann ein beliebiger Benutzer das Paket ausführen und den Dienst starten und anhalten. Im Gegensatz zu nativen Anwendungen erfordern virtuelle Anwendungen für diese Vorgänge keine Administratorenrechte.

Verwenden von ThinApp-Paketen mit Netzwerkstreaming

Jedes Netzwerkspeichergerät kann für Hunderte oder Tausende von Clientcomputern als Streamingserver verwendet werden. Siehe Abbildung 3-1.

Abbildung 3-1. Datenblockstreaming über eine Netzwerkfreigabe



Auf dem Desktop des Endbenutzers können Sie Verknüpfungen erstellen, die auf die zentral gehosteten ausführbaren Dateipakete verweisen. Wenn der Benutzer auf die Verknüpfung klickt, beginnt die Anwendung mit der Übertragung per Stream auf den Clientcomputer. Während des anfänglichen Streaming-Startvorgangs wird der Benutzer über die ThinApp-Statusleiste über den Fortschritt informiert.

So funktioniert Anwendungsstreaming mit ThinApp

Wenn Sie komprimierte ausführbare ThinApp-Dateien auf einer Netzwerkfreigabe oder einem USB-Flash-Laufwerk ablegen, werden die Inhalte der ausführbaren Datei im Blockformat per Stream an die Clientcomputer übertragen. Da eine Anwendung bestimmte Teile von Datendateien anfordert, liest ThinApp diese Informationen im komprimierten Format über das Netzwerk und verwendet dazu das Standardprotokoll für Dateifreigaben von Windows. Eine Ansicht des Prozesses erhalten Sie in Abbildung 3-2.

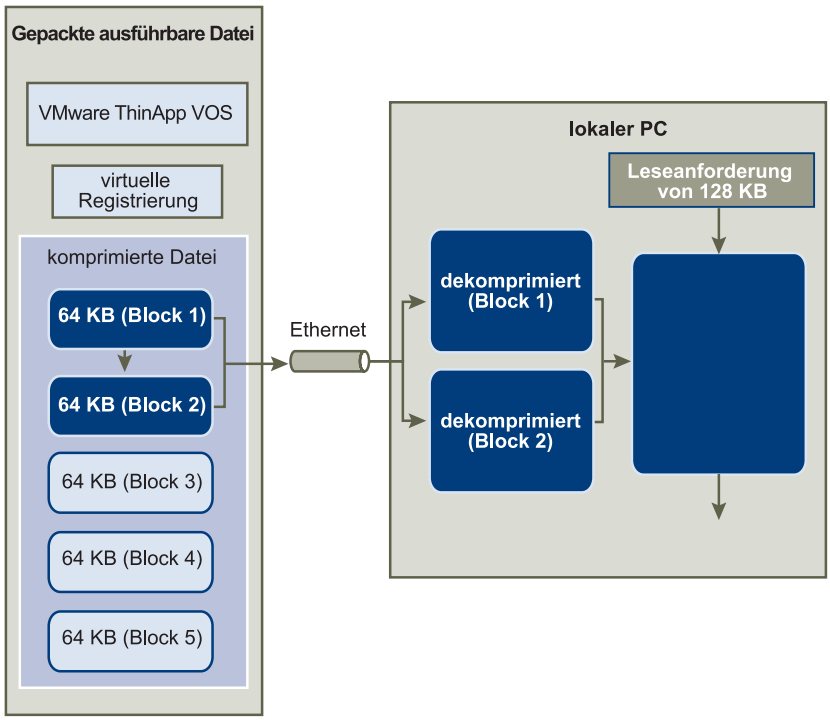
Nachdem ein Clientcomputer Daten empfangen hat, entpackt ThinApp die Daten direkt in den Arbeitsspeicher. Da ThinApp keine Daten zurück auf die Festplatte schreibt, erfolgt dieser Vorgang sehr schnell. Das Laden eines großen Pakets über das Netzwerk dauert nicht notwendigerweise lange und die Paketgröße hat keinen Einfluss auf die Startzeit einer Anwendung. Wenn Sie weitere 20 GB zu einem Paket hinzufügen, das während der Laufzeit nicht verwendet wird, wird das Paket mit der gleichen Geschwindigkeit geladen. Wenn die Anwendung geöffnet wird und 32 KB Daten von der 20 GB-Datei gelesen werden, fordert ThinApp nur 32 KB Daten an.

Der ThinApp-Laufzeitclient ist ein kleiner Bestandteil des ausführbaren Dateipakets. Wenn ThinApp den Laufzeitclient ausführt, richtet es die Umgebung ein und startet die ausführbare Zieldatei. Die ausführbare Zieldatei greift auf andere Teile der Anwendung zu, die in dem virtuellen Betriebssystem gespeichert sind. Der Laufzeitclient fängt solche Anforderungen ab und führt sie durch Laden von DLLs vom virtuellen Betriebssystem aus.

Die Ladezeit des Laufzeitclients über ein Netzwerk hinweg beträgt einige Millisekunden. Nachdem ThinApp den Laufzeitclient in den Arbeitsspeicher des Clientcomputers geladen hat, berechnet der Computer des Endbenutzers, welche Datenblöcke vom Server benötigt werden, und liest diese basierend auf der Aktivität der Anwendung.

Wenn die Anwendung anschließend Leseanforderungen für dieselben Daten durchführt, liefert der Windows-Festplatten-Cache Daten, ohne dass ein Lesevorgang auf dem Netzwerk erforderlich ist. Falls der Clientcomputer wenig Arbeitsspeicher hat, löscht Windows einen Teil des Festplatten-Caches, damit Arbeitsspeicherressourcen für andere Anwendungen zur Verfügung stehen.

Abbildung 3-2. Streaming von Anwendungen



Anforderungen und Empfehlungen für Streamingpakete

ThinApp benötigt keine spezielle Serversoftware, um die Streamingfunktion bereitstellen zu können. Jede Windows-Dateifreigabe, jedes NAS-Gerät und jede SMB-Freigabe kann diese Funktion zur Verfügung stellen. Die Datenmenge, die übertragen werden muss, bevor die Anwendung ausgeführt werden kann, ist für jede Anwendung unterschiedlich. Für Microsoft Office muss nur ein Bruchteil des Paketinhalts gestreamt werden, bevor eine Anwendung ausgeführt werden kann.

VMware empfiehlt, dass Sie ThinApp-Streaming in einer LAN-basierten Umgebung mit mindestens 100 MB-Netzwerken verwenden. Für WAN- und Internet-Bereitstellungen, bei denen häufige oder unerwartete Unterbrechungen stattfinden können, empfiehlt VMware eine der folgenden Lösungen:

- Verwenden Sie eine URL zur Bereitstellung der Anwendungen.
- Verwenden Sie eine Desktop-Bereitstellungslösung, um das Paket per Push in den Hintergrund zu bringen. Lassen Sie die Ausführung der Anwendung erst zu, nachdem das gesamte Paket heruntergeladen wurde.

Durch diese Lösungen werden Fehler verringert und Situationen beseitigt, in denen die Anwendung während eines Netzwerkausfalls nicht gestreamte Teile erfordert. Ein Unternehmen mit vielen Zweigstellen bestimmt normalerweise ein Anwendungs-Repository, das einen zentralen Freigabeordner in jeder Zweigstelle spiegelt. Mit dieser Konfiguration wird die lokale Leistung für Clientcomputer in den einzelnen Zweigstellen optimiert.

Sicherheitsempfehlungen für Streamingpakete

VMware empfiehlt, ein zentrales Freigabeverzeichnis für das Paket anzulegen und den Ordner als schreibgeschützt festzulegen. Benutzer können den jeweiligen Paketinhalt lesen, jedoch die ausführbaren Dateiinhalte nicht ändern. Wenn ein Paket von einem freigegeben Speicherort gestreamt wird, speichert ThinApp die Anwendungsänderungen in der Benutzer-Sandbox. Der Standardspeicherort für die Sandbox lautet `%AppData%\Thinstall\<Anwendungs_Name>`. Sie können den Sandbox-Speicherort während der Laufzeit oder während des Paketierens konfigurieren.

Üblicherweise erfolgt die Konfiguration derart, dass die Benutzer-Sandbox auf ein anderes zentrales Speichergerät gelegt wird. Der Benutzer kann einen beliebigen Computer verwenden und auf der zentralen Freigabe die individuellen Anwendungseinstellungen beibehalten. Wenn Pakete von einer zentralen Freigabe gestreamt werden, bleiben sie gesperrt, bis alle Benutzer die Anwendung beendet haben.

Streamen von ThinApp-Paketen über das Netzwerk

Benutzer können über das Netzwerk auf paketierte Anwendungen zugreifen.

So streamen Sie Pakete vom Netzwerk

1 Legen Sie das ThinApp-Paket an einem Speicherort ab, auf den Clientcomputer zugreifen können.

2 Senden Sie einen Link an die Benutzer, damit sie die Anwendung direkt ausführen können.

Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten

Gekapselte Anwendungen können mit anderen Komponenten interagieren, die auf dem Desktop installiert sind.

Ausführen von Einfügevorgängen

Nachfolgend erhalten Sie einen Überblick über die mit ThinApp durchführbaren Einfügevorgänge sowie die für ThinApp geltenden Einschränkungen:

- **Inhalt von systemseitig installierten Anwendungen in gekapselte Anwendungen einfügen** – Dieser Einfügevorgang ist unbegrenzt. Die virtuelle Anwendung kann alle Standard-Zwischenablageformate empfangen, wie beispielsweise Text, Grafiken und HTML. Die virtuelle Anwendung kann OLE-Objekte empfangen.
- **Einfügen von gekapselten Anwendungen in Systemanwendungen** – ThinApp konvertiert OLE-Objekte, die in virtuellen Anwendungen erstellt wurden, in native Objekte, wenn Sie sie in native Anwendungen einfügen.

Zugriff auf Drucker

Eine gekapselte Anwendung hat Zugriff auf alle Drucker, die auf dem Computer installiert sind, auf dem sie ausgeführt wird. Gekapselte Anwendungen und Anwendungen, die auf dem physischen System installiert sind, haben die gleichen Druckfähigkeiten.

Sie können ThinApp nicht zur Virtualisierung von Druckertreibern verwenden. Sie müssen die Druckertreiber manuell auf einem Computer installieren.

Zugriff auf Treiber

Eine gekapselte Anwendung hat vollständigen Zugriff auf alle Gerätetreiber, die auf dem Computer installiert sind, auf dem sie ausgeführt wird. Gekapselte Anwendungen und Anwendungen, die auf dem physischen System installiert sind, haben die gleichen Beziehungen zu Gerätetreibern. Wenn eine Anwendung einen Gerätetreiber benötigt, müssen Sie diesen Treiber separat von dem ThinApp-Paket installieren.

In manchen Fällen könnte eine Anwendung ohne einen zugehörigen Treiber funktionieren, jedoch mit bestimmten Einschränkungen. Beispielsweise installiert Adobe Acrobat einen Druckertreiber, mit dem Anwendungen unter Verwendung eines Druckmechanismus systemweit PDF-Dateien erstellen können. Wenn Sie eine gekapselte Version von Adobe Acrobat verwenden, können Sie diese zum Laden, Bearbeiten und Speichern von PDF-Dateien verwenden, ohne dass die Druckertreiber installiert werden müssen. Andere Anwendungen erkennen einen neuen Druckertreiber erst dann, wenn der Treiber installiert ist.

Zugriff auf die lokale Festplatte, den Wechseldatenträger und die Netzwerkfreigaben

Wenn Sie eine Projektstruktur erstellen, konfiguriert ThinApp die Isolationsmodi für Verzeichnisse und Unterstrukturen der Registrierung. Die Isolationsmodi steuern, für welche Verzeichnisse die Anwendung auf dem lokalen Computer Lese- und Schreibberechtigungen erhält. Überblick über die Standard-Konfigurationsoptionen:

- **Festplatte** – Ein Beispiel für eine Festplatte lautet `C:\`. Isolationsmodi, die während des Kapselungsvorgangs ausgewählt werden, haben Auswirkungen auf den Zugriff. Benutzer haben Schreibberechtigungen auf ihrem `Desktop` und im Ordner `Eigene Dateien`. Andere von der Anwendung ausgeführte Änderungen werden zur Benutzer-Sandbox weitergeleitet. Der Standard-Speicherort für die Sandbox ist das Verzeichnis `Anwendungsdaten`.
- **Wechseldatenträger** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem Wechseldatenträger.
- **Netzlaufwerke** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem vom Netzwerk zugeordneten Laufwerk.
- **UNC-Netzwerkpfade** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem UNC-Netzwerkpfad.

Zugriff auf die Systemregistrierung

Standardmäßig haben gekapselte Anwendungen Leseberechtigungen für die vollständige Systemregistrierung, wie durch die Zugriffsrechte festgelegt. Bestimmte Bereiche der Registrierung werden während des Paketerstellungsvorgangs vom System isoliert. Diese Isolation verringert Konflikte zwischen verschiedenen Versionen virtueller Anwendungen und systemseitig installierter Anwendungen. Standardmäßig speichert ThinApp alle Registrierungsänderungen von gekapselten Anwendungen in einer isolierten Sandbox und das System bleibt unverändert.

Zugriff auf Netzwerk und Sockets

Gekapselte Anwendungen haben Standardzugriff auf die Netzwerkfunktionen. Gekapselte Anwendungen können eine Bindung mit lokalen Schnittstellen eingehen und Remoteverbindungen erstellen, wenn der Benutzer Zugriffsrechte zum Ausführen dieser Vorgänge besitzt.

Verwenden von gemeinsam genutztem Arbeitsspeicher und Named Pipes

Gekapselte Anwendungen können mit anderen Anwendungen im System interagieren, indem sie gemeinsam genutzten Arbeitsspeicher, Named Pipes, Mutex-Objekte und Semaphoren verwenden.

ThinApp kann gemeinsam genutzte Arbeitsspeicherobjekte und Synchronisierungsobjekte isolieren. Durch die Isolierung werden sie für andere Anwendungen unsichtbar und andere Anwendungsobjekte werden für eine gekapselte Anwendung unsichtbar.

Verwenden von COM-, DCOM- und Out-of-Process-COM-Komponenten

Gekapselte Anwendungen können COM-Steuerungen von der virtuellen Umgebung und vom System erstellen. Wenn eine COM-Steuerung als ein Out-of-Process-COM installiert wird, wird die Steuerung als virtueller Prozess ausgeführt, wenn sie von einer gekapselten Anwendung verwendet wird. Sie können die von den gekapselten Anwendungen vorgenommenen Änderungen steuern.

Starten von Diensten

Gekapselte Anwendungen können systemseitig installierte Dienste und virtuelle Dienste starten und ausführen. Systemdienste werden in der virtuellen Umgebung ausgeführt, die die von den Diensten vorgenommenen Änderungen steuert.

Verwenden von Dateitypzuordnungen

Gekapselte Anwendungen können systemseitig installierte Anwendungen ausführen, indem sie Dateitypzuordnungen verwenden. Sie können Dateitypzuordnungen zur Registrierung des lokalen Computers hinzufügen, die auf gekapselte ausführbare Dateien für einzelne Benutzer und Rechner verweisen.

Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext

Isolationsmodi steuern den Lese- und Schreibzugriff für bestimmte Systemverzeichnisse und untergeordnete Schlüssel der Systemregistrierung.

Sie können die Isolationsmodi anpassen, um die Probleme in Tabelle 3-2 zu lösen.

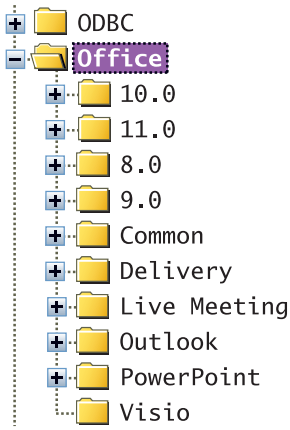
Tabelle 3-2. Beispiele für Probleme und Lösungen mithilfe von Isolationsmodi

Problem	Lösung
Eine Anwendung kann nicht ausgeführt werden, weil gleichzeitig ältere oder neuere Versionen vorhanden sind bzw. nicht ordnungsgemäß deinstalliert wurden.	Verwenden Sie den Isolationsmodus „Full (Voll)“. ThinApp verbirgt Hostcomputerdateien und Registrierungsschlüssel vor der Anwendung, wenn sich die Hostcomputerdateien in denselben Verzeichnissen und untergeordneten Schlüsseln befinden, die das Installationsprogramm der Anwendung erstellt. Für Verzeichnisse und untergeordnete Schlüssel mit dem Isolationsmodus „Full (Voll)“ erkennt die Anwendung nur virtuelle Dateien und untergeordnete Schlüssel. Alle Systemwerte, die am selben Speicherort vorhanden sind, sind für die Anwendung unsichtbar.
Eine Anwendung funktioniert nicht, weil die Benutzer sie nicht für eine Umgebung mit mehreren Benutzern konzipiert oder getestet haben. Die Anwendung kann keine Dateien und Schlüssel ändern, ohne dass dies Auswirkungen auf andere Benutzer hat.	Verwenden Sie den Isolationsmodus „WriteCopy“. ThinApp erstellt Kopien von Registrierungsschlüsseln und Dateien, die die Anwendung schreibt, und führt alle Änderungen in einer benutzerspezifischen Sandbox durch. Für Verzeichnisse und untergeordnete Schlüssel mit WriteCopy-Isolation erkennt die Anwendung die Hostcomputerdateien und virtuellen Dateien. Alle Schreibvorgänge konvertieren Hostcomputerdateien in virtuelle Dateien in der Sandbox.
Eine Anwendung funktioniert nicht, weil sie eine Schreibberechtigung für globale Speicherorte besitzt und nicht für eine gesperrte Desktop-Umgebung entworfen wurde, wie sie in Unternehmensumgebungen oder für Windows Vista üblich ist.	Verwenden Sie den Isolationsmodus „WriteCopy“. ThinApp erstellt Kopien von Registrierungsschlüsseln und Dateien, die die Anwendung schreibt, und führt alle Änderungen in einer benutzerspezifischen Sandbox durch. Für Verzeichnisse und untergeordnete Schlüssel mit WriteCopy-Isolation erkennt die Anwendung die Hostcomputerdateien und virtuellen Dateien. Alle Schreibvorgänge konvertieren Hostcomputerdateien in virtuelle Dateien in der Sandbox.

Ansicht der Auswirkung des Isolationsmodus auf die Windows-Registrierung

Abbildung 3-3 zeigt einen Bereich der Windows-Registrierung für einen Computer, auf dem ältere Microsoft Office-Anwendungen installiert sind. Microsoft Office 2003 erstellt die folgende Unterstruktur der Registrierung: `HKEY_LOCAL_MACHINE\Software\Microsoft\Office\11.0`.

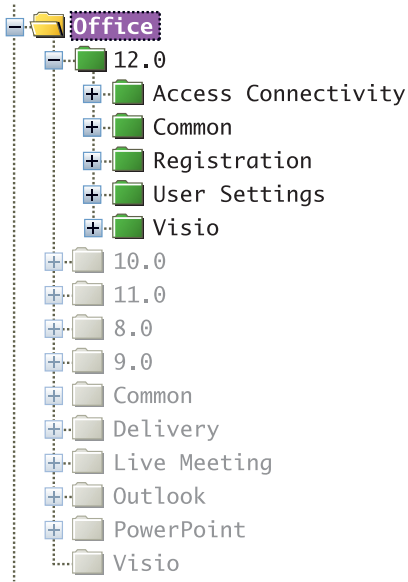
Abbildung 3-3. Die Windows-Registrierung, wie sie von Windows Regedit gesehen wird



Wenn ThinApp eine gekapselte Version von Microsoft Visio 2007 ausführt, setzt ThinApp die Unterstruktur der Registrierung `HKLM\Software\Microsoft\Office` auf den Isolationsmodus „Full (Voll)“. Mit dieser Einstellung wird verhindert, dass Microsoft Visio 2007 wegen der Registrierungseinstellungen, die möglicherweise am selben Speicherort auf dem Hostcomputer vorhanden sind, nicht funktioniert.

Abbildung 3-4 zeigt die Registrierung aus der Perspektive der gekapselten Microsoft Visio 2007-Anwendung.

Abbildung 3-4. Die Windows-Registrierung, wie sie von der gekapselten Microsoft Visio 2007-Anwendung gesehen wird



Aktualisieren und Verknüpfen von Anwendungen

4

Virtuelle Anwendungen können – abhängig vom Umfang der Änderung und den Abhängigkeiten zu anderen Anwendungen – mit unterschiedlichen Dienstprogrammen aktualisiert werden.

Dieser Abschnitt umfasst die folgenden Themen:

- „Anwendungs-Updates, die der Endbenutzer auslöst“ auf Seite 43
- „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 51
- „Automatische Anwendungs-Updates“ auf Seite 53
- „Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe“ auf Seite 54
- „Sandbox-Überlegungen für aktualisierte Anwendungen“ auf Seite 55
- „Aktualisieren der ThinApp-Version von Paketen“ auf Seite 55

Anwendungs-Updates, die der Endbenutzer auslöst

ThinApp bietet zur Aktualisierung von Anwendungen mit neuen Versionen oder neuen Komponenten die Dienstprogramme Application Sync und Application Link. Das Dienstprogramm Application Sync aktualisiert ein gesamtes Anwendungspaket. Das Dienstprogramm Application Link verwaltet gemeinsam genutzte Komponenten oder abhängige Anwendungen in separaten Paketen.

Application Sync-Updates

Mit dem Dienstprogramm Application Sync können bereitgestellte virtuelle Anwendungen auf dem neuesten Stand gehalten werden. Ist das Dienstprogramm beim Start einer Anwendung aktiviert, führt die Anwendung eine Webserverabfrage durch, um festzustellen, ob eine aktualisierte Version der ausführbaren Datei verfügbar ist. Ist ein Update verfügbar, werden die Unterschiede zwischen dem bestehenden Paket und dem neuen Paket heruntergeladen und verwendet, um eine aktualisierte Version des Pakets zu erzeugen. Bei zukünftigen Startvorgängen wird das aktualisierte Paket verwendet.

Das Dienstprogramm Application Sync ist für größere Konfigurations-Updates der Anwendung geeignet. Beispielsweise, wenn Sie ein Update auf die nächste Hauptversion von Firefox vornehmen. Remotebenutzer oder Benutzer, die nicht mit dem Unternehmensnetzwerk verbunden sind, können das Dienstprogramm Application Sync nutzen, indem Sie Update-Einstellungen in das Paket einbetten und einen Webserver verwenden, um die aktualisierte Version des Pakets zu speichern.

Verwendung von Application Sync in einer verwalteten oder nicht verwalteten Umgebung

Für virtuelle Anwendungen, die in einer verwalteten Computerumgebung automatisch aktualisiert werden, sollte Application Sync nicht verwendet werden, da das Dienstprogramm möglicherweise mit anderen Update-Funktionen Konflikte hervorruft.

Wird eine Anwendung mithilfe einer automatischen Update-Funktion aktualisiert, ist das Update in der Sandbox gespeichert. Versucht das Dienstprogramm Application Sync, die Anwendung nach einem erfolgten automatischen Anwendungs-Update zu aktualisieren, hat das in der Sandbox gespeicherte Versions-Update Vorrang vor den in der Application Sync-Version enthaltenen Dateien. Vorrang in der Reihenfolge der Dateiaktualisierungen haben die Dateien in der Sandbox, gefolgt von denen des virtuellen Betriebssystems vor denen des physischen Computers.

Verwenden Sie in einer nicht verwaltete Umgebung, in dem keinen automatischen Update der Anwendungen erfolgt, das Dienstprogramm Application Sync zum Update von Anwendungen.

Update von Firefox 2.0.0.3 auf Firefox 3 mit Application Sync

Dieses Beispiel zeigt den Update-Vorgang von Application Sync für Firefox.

Der Update-Vorgang erfordert die Modifizierung der `Package.ini`-Datei. Der `AppSyncURL`-Parameter erfordert einen URL-Pfad. ThinApp unterstützt HTTP, HTTPS und Dateiprotokolle. Informationen über alle Parameter von Application Sync finden Sie unter „Konfigurieren von Anwendungs-Updates mit Application Sync“ auf Seite 88.

So aktualisieren Sie Firefox 2.0.0.3 auf Firefox 3

1 Kapseln Sie Firefox 2.0.0.3 und Firefox 3 in getrennte Pakete.

2 Stellen Sie sicher, dass der Name des primären Datencontainers der beiden Pakete übereinstimmt.

   Der im Setup Capture-Prozess festgelegte primäre Datencontainer ist diejenige Datei, die das virtuelle Dateisystem und die virtuelle Registrierung enthält. Wenn Sie über ein Firefox 2.0.0.3-Paket, bei dem der Name des primären Datencontainers `Mozilla Firefox 2.0.0.3.exe` lautet, und ein Firefox 3-Paket verfügen, bei dem der Name des primären Datencontainers `Mozilla Firefox 3.dat` lautet, müssen Sie den Namen im `Shortcut`-Parameter in einen gemeinsamen Namen ändern. Sie können beispielsweise `Firefox.exe` als Namen verwenden.

3 Modifizieren Sie die `Package.ini`-Datei in jedem Paket.

   a Öffnen Sie die `Package.ini`-Datei, die sich im Ordner der gekapselten Anwendung befindet.

      Ein Pfad von Firefox 2.0.0.3 zur `Package.ini`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini` sein.

   b Heben Sie die Auskommentierung der Application Sync-Parameter auf, die Sie bearbeiten möchten, indem Sie das Semikolon am Anfang der Zeile entfernen.

      Um das Dienstprogramm zu aktivieren, müssen Sie die Auskommentierung vom `AppSyncURL`-Parameter aufheben.

   c Ändern Sie den Wert der Parameter und speichern Sie die Datei.

      Sie können beispielsweise eine ausführbare Datei der neuesten Firefox-Version auf ein zugeordnetes Netzlaufwerk kopieren und als Wert für den `AppSyncURL`-Parameter einen Pfad zu diesem Speicherort eingeben. Ist `Z:` das zugeordnete Laufwerk und `Firefox` der Name des Verzeichnisses, in dem die ausführbare Datei gespeichert wird, lautet ein Beispielpfad: `file:///Z:/Firefox/Firefox.exe`.

      Stellen Sie sicher, dass der Pfad `AppSyncURL` in beiden `Package.ini`-Dateien der gleiche ist und auf die aktualisierte Version verweist.

4 Doppelklicken Sie im gekapselten Anwendungsordner auf die `build.bat`-Datei, um das Anwendungspaket erneut zu erstellen.

   Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

5 Um Firefox 2.0.0.3 auf Firefox 3 zu aktualisieren, starten Sie die ausführbare Datei, wie beispielsweise `Mozilla Firefox 2.0.0.3.exe`, im `\bin`-Verzeichnis.

Wenn Sie die Anwendung vor der Ablaufzeit, die im Parameter `AppSyncExpirePeriod` der `Package.ini`-Datei eingerichtet wird, starten, lädt ThinApp das Update im Hintergrund herunter, während Sie weiterhin mit der Anwendung arbeiten. Wenn Sie die Anwendung das nächste Mal starten, sehen Sie die aktualisierte Version.

Wenn Sie die Anwendung nach Ablauf des Pakets starten, lädt ThinApp das Update im Vordergrund herunter und Sie können während dieses Zeitraums nicht mit der Anwendung arbeiten. Nach Fertigstellung des Downloads startet ThinApp die Anwendung mit der neuen Version.

Korrigieren eines Updates mithilfe von Application Sync

Wenn Sie über mehrere Download-Updates von Application Sync verfügen, beispielsweise mehrere Updates von Microsoft Office, und wenn ein bestimmtes Update sich negativ auswirkt oder entfernt werden muss, können Sie das Problem beheben.

So korrigieren Sie ein Update

Speichern Sie das richtige Update auf den Server, auf den ThinApp zugreifen kann.

Das Update wird angewendet, sobald die Anwendung das nächste Mal auf einem Clientcomputer gestartet wird.

Auswirkung von Application Sync auf ausführbare Dateien mit Einstiegspunkt

Das Dienstprogramm Application Sync aktualisiert ausführbare Dateien mit Einstiegspunkt. Angenommen, Sie stellen ein Microsoft Office 2007-Paket bereit, in dem Microsoft PowerPoint nicht enthalten ist. Der Einstiegspunkt `Microsoft Office PowerPoint 2007.exe` ist im ursprünglichen Paket nicht vorhanden. Wenn Sie das Microsoft Office 2007-Paket erneut erstellen, so dass Microsoft PowerPoint enthalten ist, und Sie das Dienstprogramm Application Sync zum Update der Clientcomputer verwenden, können die Endbenutzer auf eine ausführbare Datei mit Einstiegspunkt für Microsoft PowerPoint zugreifen.

Aktualisieren von thinreg.exe-Registrierungen mit Application Sync

Wenn Sie mithilfe von `thinreg.exe` virtuelle Anwendungen auf Ihrem System registrieren und Anwendungen mit dem Dienstprogramm Application Sync aktualisieren, können Sie Registrierungen aktualisieren, indem Sie eine Kopie von `thinreg.exe`, die sich unter `C:\Programme\VMware\VMware ThinApp` befindet, zusammen mit dem aktualisierten Paket auf dem Server speichern.

Beibehalten des primären Datencontainernamens mit Application Sync

Das Dienstprogramm Application Sync erfordert, dass der Name des primären Datencontainers und die Datei, in der die virtuellen Dateien und die Registrierungsdaten gespeichert sind, in den alten und neuen Versionen einer Anwendung identisch sind. Sie dürfen beispielsweise keine alte Version mit `Microsoft Office Excel 2003.exe` als primären Datencontainernamen verwenden, wenn die neue Version `Microsoft Office 2007.dat` als primären Datencontainernamen verwendet. Um den Namen des primären Datencontainers zu bestätigen, prüfen Sie bitte die `ReadOnlyData`-Parameter in der `Package.ini`-Datei. Weitere Informationen über den primären Datencontainer finden Sie unter „Definition von Einstiegspunkten als Verknüpfungen zur virtuellen Umgebung“ auf Seite 17.

Abschließen des Application Sync-Vorgangs, wenn die Anwendungen untergeordnete Prozesse erstellen

Wenn eine gekapselte Anwendung untergeordnete Prozesse erstellt, kann ThinApp den Application Sync-Vorgang nicht abschließen.

Angenommen Sie erstellen zum Beispiel ein Microsoft Office 2003-Paket und ein Microsoft Office 2007-Paket, modifizieren die `AppSyncURL`-Parameter in der `Package.ini`-Datei für beide Pakete und kopieren das Microsoft Office 2007-Paket auf einen Webserver und das Microsoft Office 2003-Paket auf einen Clientcomputer.

Wenn Sie das Microsoft Office 2003-Paket vor der Ablaufzeit, die im Parameter `AppSyncExpirePeriod` der `Package.ini`-Datei eingerichtet wurde, starten, kann ThinApp das Update im Hintergrund herunterladen, während Sie weiterhin mit der Anwendung arbeiten, ist jedoch nicht in der Lage, die aktualisierte Version anzuzeigen, wenn Sie die Anwendung das nächste Mal starten. Wenn Sie die Anwendung nach Ablauf des Pakets starten, ist ThinApp nicht in der Lage, das Update im Vordergrund herunterzuladen und die Anwendung erneut zu starten, nachdem der Download beendet wurde.

Microsoft Office 2003 und Microsoft Office 2007 sind Beispiele für Anwendungen, die untergeordnete Prozesse erstellen. ThinApp kann Application Sync-Updates erst dann durchführen, wenn alle untergeordneten Prozesse angehalten wurden. Sie können eine der folgenden Aufgaben vornehmen, um dieses Problem zu lösen:

- Melden Sie sich ab und melden Sie sich an dem betreffenden Rechner an, um die untergeordneten Prozesse anzuhalten.
- Erstellen Sie ein Skript, um die untergeordneten Prozesse zu beenden.

  Sie können beispielsweise ein Skript erstellen, um die untergeordneten Prozesse `ctfmon.exe` und `mdm.exe`, die mit Microsoft Office 2003 und Microsoft Office 2007 verknüpft sind, zu beenden.
- Verhindern Sie das Starten eines untergeordneten Prozesses, wie den `ctfmon.exe`-Prozess, der mit den Anwendungen Microsoft Office und Internet Explorer verknüpft ist.

Verhindern Sie das Starten des ctfmon.exe-Prozesses für Microsoft Office und Internet Explorer

Das Verhindern des Startes des `ctfmon.exe`-Prozesses erfordert Kenntnisse über die ThinApp-Sandbox und das Dienstprogramm `sbmerge.exe`. Informationen über das Dienstprogramm `sbmerge.exe` finden Sie unter „Aktualisieren von Anwendungen mit Laufzeitänderungen“ auf Seite 52.

So verhindern Sie das Starten des ctfmon.exe-Prozesses

1. Wenn Sie den Einstiegspunkt `cmd.exe` während des Kapselungsvorgangs nicht aktiviert haben, setzen Sie den Parameter `Disabled (Deaktiviert)` für den Eintrag `cmd.exe` in der `Package.ini`-Datei auf 0 und erstellen Sie das Paket mithilfe des Dienstprogramms `build.bat` neu.

   Dadurch wird im `/bin`-Verzeichnis eine ausführbare Datei für den Einstiegspunkt `cmd.exe` generiert.
2. Kopieren Sie das `/bin`-Verzeichnis im gekapselten Anwendungsverzeichnis in eine neu aufgesetzte virtuelle Maschine oder löschen Sie die Sandbox für das Microsoft Office-Paket.
3. Doppelklicken Sie auf den Einstiegspunkt `cmd.exe`.
4. Führen Sie im Windows Command Processor den Befehl `INTL.CPL` aus.
5. Klicken Sie im Register **Sprachen (Languages)** des Dialogfelds **Regional und Sprachen (Regional and Languages)** auf **Details**.
6. Aktivieren Sie im Register **Erweitert (Advanced)** des Dialogfelds **Textdienste und Eingabesprachen (Text Services and Input Languages)** das Kontrollkästchen **Erweiterte Textdienste abschalten (Turn off advanced text services)**.
7. Klicken Sie in allen geöffneten Dialogfeldern auf **OK** und schließen Sie den Windows Command Processor nicht.
8. Entfernen Sie die Registrierung der Dateien `MSIMTF.dll` und `MSCTF.dll` mithilfe des Befehls `REGSVR32.EXE/U <DLL_file>`.

   Vergleichen Sie dazu den Knowledge Base-Artikel 282599 auf der Website von Microsoft.
9. Schließen Sie den Windows Command Processor.
10. Kopieren Sie die Sandbox aus dem Paket in das Paketsystem, falls die virtuelle Maschine nicht auf demselben Computer gespeichert ist, auf dem auch ThinApp installiert ist.

    Der Standardspeicherort für die Sandbox lautet `%APPDATA%\Thinstall`.

11 Verwenden Sie von der Standard-Eingabeaufforderung auf dem Paketsystem das Dienstprogramm `sbmerge.exe`, um die aktualisierte Sandbox mit dem Paket zusammenzuführen.

   Ein Beispielbefehl lautet `SBMERGE APPLY -ProjectDir "C:\Programme\VMware \VMware ThinApp\Captures\Microsoft Office Professional 2007" -SandboxDir "%APPDATA%\Thinstall\Microsoft Office Pro 2007"`.

12 Erstellen Sie das Paket erneut und testen Sie das Paket auf einer neu aufgesetzten virtuellen Maschine, um zu bestätigen, dass der Prozess `ctfmon.exe` nicht mehr vorhanden ist.

Application Link-Updates

Das Dienstprogramm Application Link verbindet abhängige Anwendungen während der Laufzeit. Sie können Komponenten getrennt paketieren, bereitstellen und aktualisieren, anstatt alle Komponenten im selben Paket zu kapseln.

ThinApp unterstützt die gleichzeitige Kombination von bis zu 250 Paketen. Jedes Paket kann von beliebiger Größe sein.

Das Dienstprogramm Application Link ist für folgende Objekte nützlich:

- **Große gemeinsam genutzte Bibliotheken und Frameworks** – Verknüpfen der Laufzeit-Komponenten wie .NET, JRE oder ODBC-Treibern mit abhängigen Anwendungen.

  Sie können beispielsweise .NET mit einer Anwendung verknüpfen, selbst wenn der lokale Computer für die Anwendung die Installation von .NET nicht zulässt oder bereits über eine andere Version von .NET verfügt.

  Wenn Sie mehrere Anwendungen haben, die .NET erfordern, können Sie Speicherplatz sparen und ein einziges .NET-Paket erstellen und diese Anwendungen auf das .NET-Paket verweisen. Wenn Sie .NET mit einem Sicherheitsfix aktualisieren, reicht es aus, ein einzelnes Paket anstelle von mehreren Paketen zu aktualisieren.

- **Zusätzliche Komponenten und Plug-Ins** – Getrenntes Verpacken und Bereitstellen der anwendungsspezifischen Komponenten und Plug-Ins von der Basisanwendung.

  Sie können beispielsweise Adobe Flash Player oder Adobe Reader von einer Firefox-Basisanwendung trennen und die Komponenten verknüpfen.

  Sie können ein einziges virtualisiertes Microsoft Office-Paket für alle Benutzer sowie individuelle zusätzliche Komponenten für einzelne Benutzer bereitstellen.

  Wenn Sie Microsoft Office kapseln und versuchen, in der virtuellen Umgebung von Microsoft Outlook auf einen Anhang zuzugreifen, können Sie Microsoft Office so einrichten, dass das Programm ein verknüpftes Adobe Reader-Paket auf dem Netzwerk erkennt, falls Adobe Reader innerhalb der unmittelbaren virtuellen oder physischen Umgebung nicht verfügbar ist.

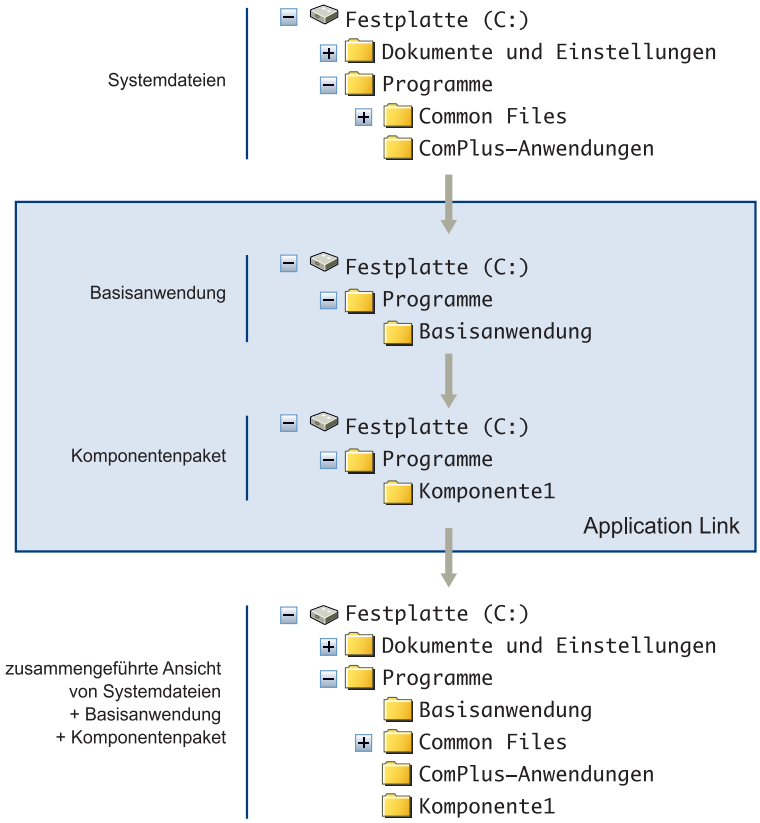
- **Hotfixes und Service Packs** – Verknüpfen der Updates mit einer Anwendung und Rollback auf eine frühere Version, wenn Benutzer mit der neuen Version auf wesentliche Probleme stoßen. Sie können kleinere Patches für Anwendungen als Einzeldatei bereitstellen und die Anzahl erforderlicher Rollbacks verringern.

  Application Link bietet Einsparungen bei der Bandbreite. Wenn Sie beispielsweise Microsoft Office 2007 Service Pack 1 haben und ohne Application Link auf Service Pack 2 aktualisieren möchten, würden Sie mit der Bereitstellung eines neuen Office 2007 Service Pack 2-Pakets 1,5 GB an Daten pro Computer übertragen. Application Link überträgt ausschließlich die Updates und nicht das gesamte Paket auf die Computer.

Ansicht der Anwendung unter Verwendung von Application Link

Abbildung 4-1 zeigt die laufende Anwendung mit einer zusammengeführten Ansicht des Systems, der Basisanwendung und aller verknüpften Komponenten. Dateien, Registrierungsschlüssel, Dienste, COM-Objekte und Umgebungsvariablen von abhängigen Paketen sind in der Basisanwendung sichtbar.

Abbildung 4-1. Ansicht des Systems, der Basisanwendung und aller verknüpften Komponenten unter Verwendung von Application Link



Verknüpfen einer Basisanwendung mit Microsoft .NET Framework

Betrachten Sie dieses Workflow-Beispiel für das Verknüpfen einer Basisanwendung, `MyApp.exe`, mit einem getrennten Paket, das Microsoft .NET 2.0 Framework enthält. Stellen Sie sicher, dass die Kapselung der Basisanwendung Microsoft .NET 2.0 Framework nicht einschließt. Informationen über die Kapselung einer Anwendung finden Sie in Kapitel 2, „Kapselung von Anwendungen“, auf Seite 15.

Informationen über erforderliche und optionale Application Link-Parameter und -Formate in der `Package.ini`-Datei finden Sie unter „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 86.

So verknüpfen Sie eine Anwendung mit Microsoft .NET

1 Kapseln Sie die Installation von .NET 2.0 Framework.

   Während des Kapselungsprozesses müssen Sie mindestens einen Einstiegspunkt auswählen, auf den Benutzer Zugriff haben.

2 Benennen Sie die von ThinApp erzeugte `.exe`-Datei in eine `.dat`-Datei um.

   Durch diese Umbenennung wird verhindert, dass Benutzer die Anwendung aus Versehen ausführen.

   Der Name, den Sie für die `.dat`-Datei auswählen, ist unbedeutend, da die Benutzer die Datei nicht direkt ausführen. Verwenden Sie beispielsweise `dotnet.dat`.

3 Speichern Sie das .NET-Projekt unter `C:\Captures\dotnet`.

4 Kapseln Sie die Basisanwendung, indem Sie dasselbe physische System oder dieselbe virtuelle Maschine verwenden, auf dem/der .NET Framework bereits installiert ist.

5 Speichern Sie das Projekt unter `C:\Captures\MyApp`.

6 Öffnen Sie die `Package.ini`-Datei, die sich im gekapselten Anwendungsordner befindet, für die Basisanwendung.

7 Aktivieren Sie den `RequiredAppLinks`-Parameter für die Basisanwendung, indem Sie nach dem Eintrag `[BuildOptions]` folgende Zeile einfügen:

```
RequiredAppLinks=dotnet.dat
```

Application Link-Parameter müssen sich auf den primären Datencontainer der Anwendung, zu der Sie einen Link erstellen möchten, beziehen. Der Bezug auf `.exe`-Verknüpfungsdateien ist nicht möglich, da diese Dateien keine Anwendungen, Dateien oder Registrierungsschlüssel enthalten.

8 Erstellen Sie Pakete von .NET 2.0 und Basisanwendung erneut.

a Doppelklicken Sie auf die Datei `build.bat` unter `C:\Captures\MyApp`.

b Doppelklicken Sie auf die Datei `build.bat` unter `C:\Captures\dotnet`.

Durch Ausführen dieser Batchdateien werden getrennte ThinApp-Pakete erstellt.

9 Stellen Sie diese Anwendungen auf einem Endbenutzer-Desktop unter `C:\Programme\MyApp` bereit.

a Kopieren Sie `C:\Captures\MyApp\bin\MyApp.exe` in `\\<Endbenutzer_Desktop>\<Programme_Freigabe>\MyApp\MyApp.exe`.

b Kopieren Sie `C:\Captures\dotnet\bin\cmd.exe` in `\\<Endbenutzer_Desktop>\<Programme_Freigabe>\MyApp\dotnet.dat`.

Einrichten verschachtelter Links mit Application Link

Mit dem Dienstprogramm Application Link unterstützt ThinApp verschachtelte Links. Wenn beispielsweise Microsoft Office einen Link zu einem Service Pack herstellt und das Service Pack einen Link zu einem Hotfix bietet, unterstützt ThinApp alle diese Abhängigkeiten.

Dieser Vorgang bezieht sich auf AppA, die AppB erfordert und AppB, die AppC erfordert. Für den Vorgang wird folgende Ordnerstruktur angenommen:

- `C:\AppFolder\AppA\AppA.exe`
- `C:\AppFolder\AppB\AppB.exe`
- `C:\AppFolder\AppC\AppC.exe`

Informationen über das Einrichten der erforderlichen und optionalen Application Link-Parameter bei diesem Verfahren finden Sie unter „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 86.

So richten Sie verschachtelte Links ein

1 Kapseln Sie Anwendung A.

2 Legen Sie in der `Package.ini`-Datei Anwendung B als erforderlichen oder optionalen Anwendungslink fest.

Fügen Sie beispielsweise `RequiredLinks=\AppFolder\AppB\AppB.exe` zur Datei hinzu.

3 Kapseln Sie Anwendung B.

4 Legen Sie in der `Package.ini`-Datei für Anwendung B die Anwendung C als erforderlichen oder optionalen Anwendungslink fest.

Fügen Sie beispielsweise `RequiredLinks=\AppFolder\AppC\AppC.exe` zur Datei hinzu.

5 Kapseln Sie Anwendung C.

Wenn Sie Anwendung A starten, kann das Programm auf die Dateien und Registrierungsschlüssel von Anwendung B zugreifen und Anwendung B kann auf die Dateien und Registrierungsschlüssel von Anwendung C zugreifen.

Beeinflussen der Isolationsmodi mit Application Link

ThinApp lädt mit dem Start der Anwendung eine Application Link-Ebene und führt Registrierungseinträge und Dateisystemverzeichnisse zusammen. Wenn ThinApp einen Registrierungsschlüssel oder Dateisystemverzeichnis ermittelt, der im bereits zusammengeführten Hauptpaket bzw. in der Hauptebene nicht vorhanden war, verwendet ThinApp den in der geladenen Ebene festgelegten Isolationsmodus. Ist der Registrierungsunterschlüssel oder das Dateisystemverzeichnis in dem Hauptpaket und einer Ebene vorhanden, die bereits zusammengeführt wurde, dann verwendet ThinApp den am meisten einschränkenden Isolationsmodus, der in einer der Ebenen oder im Hauptpaket festgelegt wurde. Die Reihenfolge der Isolationsmodi lautet „Full“, „WriteCopy“ und „Merged“, wobei für „Full“ die meisten und für „Merged“ die wenigsten Einschränkungen gelten.

Auswirkungen von PermittedGroups auf verknüpfte Pakete

Wenn Sie zwei Anwendungen verknüpfen und einen Wert für den `PermittedGroups`-Parameter festlegen, muss das Benutzerkonto, das zum Starten der Anwendung verwendet wird, mindestens zu einer der Active Directory-Gruppen für diesen Parameter in den `Package.ini`-Dateien beider Anwendungen gehören. Informationen über den `PermittedGroups`-Parameter finden Sie unter „Konfigurieren von Berechtigungen“ auf Seite 65.

Sandbox-Änderungen für eigenständige und verknüpfte Pakete

Sandbox-Änderungen von verknüpften Paketen sind für die ausführbare Basisdatei nicht sichtbar. Sie können zum Beispiel Acrobat Reader als eigenständiges virtuelles Paket installieren und als mit der Basisanwendung Firefox verknüpftes Paket. Wenn Sie Acrobat Reader als eigenständige Anwendung starten, indem Sie das virtuelle Paket ausführen und Änderungen in den Voreinstellungen vornehmen, speichert ThinApp die Änderungen in der Sandbox für Acrobat Reader. Wenn Sie Firefox starten, kann Firefox diese Änderungen nicht erkennen, weil Firefox eine eigene Sandbox hat. Beim Öffnen einer `PDF`-Datei mit Firefox werden die Änderungen der Voreinstellungen, die in der eigenständigen Anwendung Acrobat Reader vorhanden sind, nicht angezeigt.

Importierreihenfolge für verknüpfte Pakete

ThinApp importiert verknüpfte Anwendungen entsprechend der Reihenfolge der Anwendungen im `RequiredAppLinks`- oder `OptionalAppLinks`-Parameter. Wenn einer der Parameter ein Platzhalterzeichen festlegt, durch das das Importieren von mehr als einer Datei ausgelöst wird, entscheidet die alphabetische Reihenfolge darüber, welches Paket zuerst importiert wird.

Der `OptionalAppLinks`-Parameter wird möglicherweise als `OptionalAppLinks=a.exe;b.exe;plugins\*.exe` angezeigt.

Bei Verwendung von `a.exe` und `b.exe` als Beispiel für ausführbare Dateien importiert ThinApp verknüpfte Pakete in der folgenden Reihenfolge:

- Basisanwendung
- `a.exe`
- `b.exe`
- Plug-Ins, geladen in alphabetischer Reihenfolge
- Verschachtelte Plug-Ins für `a.exe`
- Verschachtelte Plug-Ins für `b.exe`
- Verschachtelte Plug-Ins für den ersten Satz der Plug-Ins in dieser Liste

Informationen über verschachtelte Links finden Sie unter „Einrichten verschachtelter Links mit Application Link“ auf Seite 49.

Datei- und Registrierungskollisionen in verknüpften Paketen

Enthalten die Basisanwendung und ein mit der Basisanwendung verknüpftes Paket Datei- oder Registrierungseinträge am selben Speicherort, dann tritt eine Kollision auf. Geschieht dies, entscheidet die Reihenfolge der Importe darüber, welches Paket Priorität hat. In solchen Fällen hat das zuletzt importierte Paket Priorität und die Datei- oder Registrierungsinhalte dieses Pakets sind für die laufenden Anwendungen sichtbar.

VBScript-Kollisionen in verknüpften Paketen

VBScript-Namenskollisionen können Skripte in anderen importierten Paketen an der Ausführung hindern. Wenn Sie Pakete mit Application Link verknüpfen und diese Pakete Skripte mit dem gleichen Namen besitzen, legt ThinApp die VBScripts von den verlinkten Paketen in einen gemeinsamen Datenpool. Bei Skripten mit dem gleichen Namen führt ThinApp das Skript von dem letzten importierten Paket aus und ignoriert das andere Skript.

Ein Basispaket enthält möglicherweise die Dateien `a.vbs` und `b.vbs` und ein abhängiges Paket enthält möglicherweise die Dateien `b.vbs` und `c.vbs`. Da zwischen den beiden `b.vbs`-Dateien eine Namenskollision besteht, überschreibt das VBScript in dem letzten importierten Paket, das in einem `RequiredAppLinks`- oder `OptionalAppLinks`-Parameter spezifiziert wurde, alle früher importierten Skripte mit dem gleichen Namen. In diesem Fall fasst ThinApp den Datenpool von vier `.vbs`-Dateien mit der Datei `a.vbs` vom Basispaket und den Dateien `b.vbs` und `c.vbs` von dem abhängigen Paket in einem gemeinsamen Datenpool zusammen.

VBScript-Funktionsreihenfolge in verknüpften Paketen

In einem Datenpool von VBScripts, die mit Application Link verknüpft sind, werden die Funktionen in den Hauptbereichen der Skripte in alphabetischer Reihenfolge der Skriptnamen ausgeführt. Die ThinApp-Rückruffunktionen in den Skripten werden in umgekehrter alphabetischer Reihenfolge der Skriptnamen im Datenpool ausgeführt.

Speichern mehrerer Versionen einer verknüpften Anwendung im selben Verzeichnis

Wenn das Verzeichnis ein verknüpftes Paket enthält und Sie eine aktualisierte Version des verknüpften Pakets zum selben Verzeichnis hinzufügen, dann erkennt das Dienstprogramm Application Link die aktualisierte Version und verwendet sie.

Verwenden von Application Sync für eine Basisanwendung und verknüpfte Pakete

Wenn Sie Application Link verwenden, um Pakete mit einem Basispaket zu verknüpfen, und das Basispaket starten, kann Application Sync nur das Basispaket aktualisieren. Wenn Sie zum Beispiel ein Microsoft Office 2007-Paket mit Application Sync-Einträgen in der `Package.ini`-Datei und ein Adobe Reader-Paket mit Application Sync-Einträgen in der `Package.ini`-Datei erstellen, und Application Link verwenden, um die beiden Pakete zu verknüpfen und dann Microsoft Office 2007 starten, aktualisiert Application Sync nur Microsoft Office 2007. Sie können sowohl Microsoft Office 2007 als auch Adobe Reader aktualisieren, indem Sie jede Anwendung einzeln starten.

Wenn Sie nicht alle Anwendungen aktualisieren und eine Basisanwendung mit einem nicht mehr gültigen Plug-In verknüpfen, kann die Basisanwendung das Plug-In dennoch laden und verwenden.

Anwendungs-Updates, die der Administrator auslöst

ThinApp bietet Administratoren die Dienstprogramme `AppSync.exe` und `sbmerge.exe`.

Das Dienstprogramm `AppSync.exe` erzwingt auf einem Clientcomputer ein Update von Application Sync.

Das Dienstprogramm `sbmerge.exe` erstellt inkrementelle Updates der Anwendungen. Ein Administrator kann das Dienstprogramm zum Beispiel verwenden, um ein Plug-In für Firefox zu integrieren oder um die Startseite einer Website auf eine neue Standardwebsite zu verweisen.

Erzwingen eines Updates von Application Sync auf Clientcomputern

Sie können mithilfe des Befehls `AppSync` ein Update von Application Sync auf einem Clientcomputer erzwingen. Möglicherweise möchten Sie ein Paket aktualisieren, das an einem Speicherort gespeichert wird, für den Standardbenutzer keinen Schreibzugriff haben. In dieser Situation können Sie nicht die Application Sync-Parameter verwenden, um beim Start einer Anwendung nach Updates zu suchen, da die Benutzer nicht über die erforderlichen Rechte zum Update des Pakets verfügen. Sie können ein tägliches Ausführen von `AppSync.exe` über ein Konto mit ausreichenden Berechtigungen planen. Die Application Sync-Parameter, wie `AppSyncUpdateFrequency`, in der `Package.ini`-Datei haben keine Auswirkungen auf den Befehl `AppSync`.

Verwenden Sie zum Erzwingen eines Application Sync-Updates den Befehl `AppSync <Anwendung_Sync_URL> <Ausführbarer_Dateipfad>`. Der Wert der URL ist derselbe wie die Application Sync-URL in der `Package.ini`-Datei und der Pfad der ausführbaren Datei ist der Pfad zur ausführbaren Datei, die aktualisiert werden soll.

Aktualisieren von Anwendungen mit Laufzeitänderungen

Das Dienstprogramm `sbmerge.exe` führt Laufzeitänderungen, die in der Sandbox der Anwendung aufgezeichnet werden, in ein ThinApp-Projekt zusammen. Ein typischer Workflow für dieses Dienstprogramm umfasst folgende Aufgaben:

- Kapseln einer Anwendung.
- Erstellen der Anwendung mit der `build.bat`-Datei.
- Ausführen einer gekapselten Anwendung und Anpassung der Einstellungen und der virtuellen Umgebung. ThinApp speichert die Änderungen in der Sandbox.
- Ausführen des Dienstprogramms `sbmerge.exe`, um die Registrierungs- und Dateisystemänderungen aus der Sandbox im ThinApp-Projekt zusammenzuführen.
- Erneutes Erstellen der gekapselten Anwendung mit der `build.bat`-Datei.
- Bereitstellen der aktualisierten Anwendung.

Sandbox-Änderungen mit Firefox zusammenführen

Dieses Verfahren für das Dienstprogramm `sbmerge.exe` verwendet Firefox 2.0.0.3 als Beispiel für eine gekapselte Anwendung.

So führen Sie Sandbox-Änderungen mit Firefox 2.0.0.3 zusammen

1 Kapseln Sie Firefox 2.0.0.3.

2 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

3 Erstellen Sie für den Sandbox-Speicherort ein `Thinstall`-Verzeichnis im `bin`-Verzeichnis.

4 Starten Sie Firefox und nehmen Sie eine Einstellungsänderung vor.

Ändern Sie beispielsweise die Startseite.

5 Navigieren Sie von der Befehlszeile zum Verzeichnis, in dem der ThinApp-Projektordner gespeichert ist.

Navigieren Sie zum Beispiel zu `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3`.

6 Geben Sie in der Befehlszeile den Befehl `"C:\Programme\VMware\VMware ThinApp\sbmerge" Print` ein.

ThinApp druckt die Änderungen, die sich durch Verwenden der gekapselten Anwendung auf den Sandbox-Ordner auswirkten.

7 Geben Sie in der Befehlszeile den Befehl `"C:\Programme\VMware\VMware ThinApp\sbmerge" Apply` ein.

ThinApp leert den Thinstall-Ordner und führt die Sandbox-Änderungen mit der Anwendung zusammen.

sbmerge.exe-Befehle

Der Befehl `sbmerge.exe Print` zeigt Sandbox-Änderungen an und nimmt weder an der Sandbox noch am ursprünglichen Projekt Änderungen vor.

Der Befehl `sbmerge.exe Apply` führt die Änderungen aus der Sandbox mit dem ursprünglichen Projekt zusammen. Dieser Befehl aktualisiert die Projektregistrierung und das Dateisystem gemäß den Änderungen und löscht das Sandbox-Verzeichnis.

Verwendung

```
"C:\Programme\VMware\VMware ThinApp\sbmerge" Print [<Optionale_Parameter>]
"C:\Programme\VMware\VMware ThinApp\sbmerge" Apply [<Optionale_Parameter>]
```

Optionale Parameter

Die optionalen `sbmerge.exe`-Parameter legen die Projekt- und Sandbox-Pfade fest und blockieren Statusmeldungen und das Zusammenführen der Sandbox-Dateien.

Parameter	Beschreibung
`–ProjectDir <Projekt_Pfad>`	Verwenden Sie den absoluten oder relativen Pfad zum Projektverzeichnis mithilfe des `–ProjectDir <Projekt_Pfad>`-Parameters, wenn Sie den `sbmerge.exe`-Befehl von einem Speicherort außerhalb des Projektordners aufrufen. Ein Beispielbefehl lautet `"C:\Programme\VMware\VMware ThinApp\sbmerge" Print –ProjectDir "C:\<Projektordner_Pfad>""`.
`–SandboxDir <Sandbox_Pfad>`	Wenn Sie eine gekapselte Anwendung starten, sucht sie in einem bestimmten Ordner nach der Sandbox. Siehe „Suchreihenfolge für die Sandbox“ auf Seite 99. Verwenden Sie den `–SandboxDir <Sandbox_Pfad>`-Parameter, um den Speicherort festzulegen, wenn Sie einen benutzerdefinierten Ordner für die Sandbox verwenden.
`–Quiet`	Blockiert das Drucken der Statusmeldungen.
`–Exclude <Ausgeschlossene_Datei>.ini`	Unterbindet das Zusammenführen bestimmter Dateien oder Registrierungseinträge aus der Sandbox. Sie können eine `.ini`-Datei festlegen, um den Inhalt, der ausgeschlossen werden soll, zu bestimmen. Diese Datei enthält getrennte Abschnitte, um Dateien festzulegen, wie `FileSystemIgnoreList` und `RegistryIgnoreList`. Das Dienstprogramm `sbmerge.exe` verwendet standardmäßig die `snapshot.ini`-Datei im Installationsordner von ThinApp, um bestimmte Inhalte von der Zusammenführung auszuschließen. Diese Option ermöglicht es, eine weitere `.ini`-Datei festzulegen, um zusätzliche Inhalte auszuschließen.

Automatische Anwendungs-Updates

Kann eine Anwendung automatisch aktualisiert werden, funktioniert ihr Update-Mechanismus mit ThinApp. Lädt die Anwendung das Update herunter und führt einen Installationsdienstprogramm oder ein Patch-Programm aus, geschieht dies innerhalb der virtuellen Umgebung und ThinApp speichert die Änderungen aus der Update-Software in der Sandbox. Startet die Anwendung neu, verwendet sie die Version der ausführbaren Datei in der Sandbox und nicht die ausführbare Datei aus dem ursprünglichen Paket.

Wenn Sie zum Beispiel Firefox 1.5 kapseln, fordert der Auto-Update-Mechanismus möglicherweise zum Upgrade auf Firefox 2.0 auf. Wenn Sie mit dem Upgrade fortfahren, lädt die Anwendung die Updates herunter, schreibt die Updates in die Sandbox und fordert zum Neustart der Anwendung auf. Führen Sie die gekapselte Anwendung erneut aus, wird Firefox 2.0 gestartet. Wenn Sie die Sandbox löschen, kehrt Firefox zu Version 1.5 zurück.

Verwenden Sie das Dienstprogramm `sbmerge.exe`, um Änderungen zusammenzuführen, die ein Auto-Update-Mechanismus mit dem ursprünglichen Paket vornimmt, damit eine aktualisierte ausführbare Datei erstellt wird. Siehe „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 51.

ANMERKUNG Deaktivieren Sie die Auto-Update-Funktionen der Anwendung, wenn Sie das Dienstprogramm Application Sync verwenden, um Anwendungs-Updates vorzunehmen. Siehe „Verwendung von Application Sync in einer verwalteten oder nicht verwalteten Umgebung“ auf Seite 43.

Dynamische Updates ohne Administratorrechte

Sie können Anwendungen dynamisch aktualisieren, ohne dass hierzu Administratorrechte erforderlich sind. Zum Beispiel müssen .NET-basierte Anwendungen, die als Teil ihres Update-Vorgangs neue DLL-Dateien aus dem Internet herunterladen, die `ngen.exe`-Datei ausführen, um Native-Image-Assemblies für den Startvorgang zu generieren. Unter normalen Umständen schreibt die `ngen.exe`-Datei in `HKLM` und `C:\WINDOWS`, auf die beiden nur mit Administratorkonten zugegriffen werden kann. Mit ThinApp kann die `ngen.exe`-Datei Native-Image-Assemblies auf Gastbenutzerkonten installieren; die Änderungen werden jedoch in einem benutzerspezifischen Verzeichnis gespeichert.

Sie können das Paket auf einem zentralen Computer aktualisieren und die Änderungen als neue gekapselte ausführbare Datei auf Clientcomputer oder zentrale Netzwerkfreigaben verschieben. Verwenden Sie eine der folgenden Optionen, um Updates anzuwenden:

- Im Verlauf des Setup Capture-Prozesses.
- Innerhalb der virtuellen Umgebung.

  Anwendungen mit Auto-Update-Funktionen können aktualisiert werden. Ist das Update eine `patch.exe`-Datei, kann das Patch-Programm in der virtuellen Umgebung und von einem `cmd.exe`-Dateieinstiegspunkt aus ausgeführt werden. Änderungen treten während automatischer oder manueller Updates in der Sandbox auf, um durch Löschen der Sandbox die Rückkehr zur ursprünglichen Version zu ermöglichen.

  Wenn Sie in der virtuellen Umgebung Patches auf einen zentralen Computer für die Paketierung anwenden, können Sie das Dienstprogramm `sbmerge.exe` verwenden, um die Sandbox-Änderungen, die durch das Update in der Anwendung vorgenommen wurden, zusammenzuführen. Siehe „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 51.
- Im gekapselten Projekt.

  Zum Update kleiner Sätze an Dateien oder Registrierungsschlüsseln können Sie die Dateien im gekapselten Projekt ersetzen. Dieser Ansatz ist für Softwareentwickler nützlich, die ThinApp-Builds in ihren Workflow integrieren.

Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe

ThinApp ermöglicht Ihnen das Upgrade oder das Rollback einer Anwendung, die auf einer Netzwerkfreigabe für mehrere Benutzer ausgeführt wird. Das Upgrade beginnt, wenn der Benutzer die Anwendung beendet und dann erneut startet. In Terminalserver-Umgebungen können mehrere Benutzer während des Übergangszeitraums verschiedene Versionen gleichzeitig ausführen.

Dateisperren

Durch Starten einer Anwendung wird das ausführbare Dateipaket gesperrt. Sie können die Anwendung weder ersetzen, noch löschen oder verschieben. Diese Dateisperre gewährleistet, dass jedem Computer oder Benutzer, der auf eine bestimmte Version einer Anwendung zugreift, diese Version weiterhin zur Verfügung steht, solange die Anwendungsprozesse und Unterprozesse ausgeführt werden.

Wenn Sie eine Anwendung für viele Benutzer an einem zentralen Speicherort speichern, verhindert diese Dateisperre, dass Administratoren eine paketierte ausführbare Datei durch eine neue Version ersetzen, bevor sämtliche Benutzer die Anwendung beendet und ihre Sperren aufgehoben haben.

Upgrade einer laufenden Anwendung

Sie können eine neue Version einer Anwendung mit einer höheren Dateinamenerweiterung, wie .1 oder .2, in ein bestehendes Bereitstellungsverzeichnis kopieren. Bei diesem Vorgang wird Firefox als Beispielanwendung verwendet.

Verknüpfungen müssen nicht aktualisiert werden.

So aktualisieren Sie eine laufende Anwendung

1 Stellen Sie die ursprüngliche Version der Anwendung, wie `Firefox.exe`, bereit.

2 Kopieren Sie die Anwendung auf eine zentrale Freigabe unter `\\<Server>\<Freigabe>\Firefox.exe`.

Ein Beispielspeicherort ist `C:\Programme\Firefox\Firefox.exe`.

3 Erstellen Sie eine Desktop- oder **Start**menüverknüpfung zum Benutzer-Desktop, die unter `\\<Server>\<Freigabe>\Firefox.exe` auf den Speicherort einer gemeinsam genutzten Datei zeigt.

Annahme: Zwei Benutzer starten `Firefox.exe` und sperren die Anwendung.

4 Kopieren Sie die aktualisierte Version der `Firefox.exe`-Datei auf die zentrale Freigabe unter `\\<Server>\<Freigabe>\Firefox.1`.

Sind Sie ein neuer Benutzer, startet ThinApp die Anwendung mit den neuen Paketdaten von `Firefox.1`. Sind Sie ein Benutzer, der mit der ursprünglichen Version arbeitet, können Sie die neue Version sehen, nachdem Sie die Anwendung beendet und die Anwendung neu gestartet haben.

5 Wenn Sie ein neueres Update von Firefox bereitstellen müssen, speichern Sie sie mit einer höheren Versionsnummer am Ende in dasselbe Verzeichnis.

6 Kopieren Sie Version 2.0 von `Firefox.exe` auf einen zentralen, gemeinsam genutzten Server unter `\\<Server>\<Freigabe>\Firefox.2`

Nachdem die Sperre von `Firefox.1` aufgehoben wurde, können Sie die Version löschen, aber die `Firefox.exe`-Datei sollte bestehen bleiben, da die Benutzerverknüpfungen weiterhin auf diese Datei zeigen. ThinApp benutzt stets den Dateinamen mit der höchsten Versionsnummer. Wenn Sie ein Rollback auf eine frühere Version vornehmen müssen und die neueste Version immer noch gesperrt ist, kopieren Sie die alte Version, sodass sie die höchste Versionsnummer trägt.

Sandbox-Überlegungen für aktualisierte Anwendungen

Wenn Sie eine Anwendung aktualisieren, können Sie steuern, ob die Benutzer weiterhin ihre früheren Einstellungen verwenden können, indem Sie den Sandbox-Namen in der `Package.ini`-Datei konsistent halten. Sie können Benutzer daran hindern, eine ältere Sandbox mit einer aktualisierten Anwendung zu verwenden, indem Sie die aktualisierte Anwendung mit einem neuen Namen für die Sandbox verpacken. Beim ersten Start der aktualisierten Anwendung wird die Sandbox mit dem neuen Namen erstellt.

Aktualisieren der ThinApp-Version von Paketen

Sie können das Dienstprogramm `relink.exe` verwenden, um ein vorhandenes Paket oder eine Paketreihe auf die neueste Version von ThinApp zu aktualisieren. Obwohl Sie die neueste Version von ThinApp installieren und das Dienstprogramm `build.bat` ausführen können, um jedes Zielpaket mit der neuesten ThinApp-Version neu zu erstellen, bietet das Dienstprogramm `relink.exe` eine schnellere Methode, die ThinApp-Version vorhandener Pakete zu aktualisieren. Mit der Aktualisierung Ihres Pakets genießen Sie die Vorteile der neuesten Funktionen oder Supportverbesserungen von ThinApp.

Relink-Beispiele

Das Dienstprogramm `relink.exe` verfügt über eine optionale `–Recursive`-Kennzeichnung und kann ein einzelnes oder mehrere Pakete anvisieren.

```
relink [–Recursive] <target> [<target> ...]
```

Sie können beispielsweise ein Adobe Reader-Paket auf die neueste installierte ThinApp-Version aktualisieren.

```
relink AdobeReader.exe
```

Das Dienstprogramm `relink.exe` kann mit Platzhaltern arbeiten.

```
relink *.exe *.dat
```

Das Dienstprogramm `relink.exe` kann zur Verarbeitung sämtlicher ThinApp-Dateien in einem Verzeichnis die Verzeichnisnamen verwenden.

```
relink C:MyPackages
```

Wenn Sie die `–Recursive`-Option festlegen, verarbeitet das Dienstprogramm `relink.exe` alle ThinApp-Dateien im Verzeichnis und sämtliche Unterverzeichnisse. Diese Option ist nur für die Verwendung von Verzeichnisnamen bestimmt.

Enthält der Zielname Leerzeichen, müssen doppelte Anführungszeichen verwendet werden.

```
relink "Microsoft Office Professional 2007.dat"
```

Konfigurieren von Paketparametern 5

Fortgeschrittene Anwender können die Parameter der virtuellen Anwendung außerhalb des Kapselungsprozesses anpassen. Parameter können sich auf die Konfiguration von Build-Optionen auswirken, darunter die Einstellungen für MSI, Aktualisierungen und Einstiegspunkte.

Die `Package.ini`-Datei befindet sich im Projektordner und enthält Parameter zur Konfiguration einer gekapselte Anwendung während des Build-Prozesses. Der Setup Capture-Assistent richtet die ursprünglichen Einstellungen für einige der Parameter in der Datei `Package.ini` ein. Damit die Parameteränderungen wirksam werden, müssen Sie die `Package.ini`-Datei speichern und das Projekt erstellen.

Dieser Abschnitt umfasst die folgenden Themen:

- „Package.ini-Dateistruktur“ auf Seite 58
- „Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden“ auf Seite 58
- „Konfigurieren der ThinApp-Laufzeit“ auf Seite 58
- „Konfigurieren des Isolationsmodus“ auf Seite 60
- „Konfigurieren von Datei- und Protokollzuordnungen“ auf Seite 62
- „Konfigurieren der Build-Ausgabe“ auf Seite 63
- „Konfigurieren von Berechtigungen“ auf Seite 65
- „Konfigurieren von Objekten und DLL-Dateien“ auf Seite 67
- „Konfigurieren von Dateispeicher“ auf Seite 72
- „Konfigurieren von Prozessen und Diensten“ auf Seite 75
- „Konfigurieren von Größen“ auf Seite 77
- „Konfigurieren der Protokollierung“ auf Seite 80
- „Konfigurieren von Versionen“ auf Seite 81
- „Konfigurieren von Gebietsschemata“ auf Seite 82
- „Konfigurieren von einzelnen Anwendungen“ auf Seite 83
- „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 86
- „Konfigurieren von Anwendungs-Updates mit Application Sync“ auf Seite 88
- „Konfigurieren von MSI-Dateien“ auf Seite 91
- „Konfigurieren von Sandbox-Speicher und Bestandsnamen“ auf Seite 96

Package.ini-Dateistruktur

Die Struktur der Datei `Package.ini` enthält sowohl Abschnitte, die für alle Anwendungen gelten, als auch Abschnitte, die nur für bestimmte Anwendungen gelten.

Die meisten Parameter müssen unter einer bestimmten Abschnittsüberschrift angegeben werden. Die Datei `Package.ini` enthält folgende Überschriften:

- `[BuildOptions]`
- `[<Anwendung>.exe]`
- `[FileList]`
- `[Compression]`
- `[Isolation]`

Der Abschnitt `[BuildOptions]` der `Package.ini`-Datei gilt für alle Anwendungen. Die einzelnen Anwendungen übernehmen diese Parameter, sofern keine anwendungsspezifischen Einträge Vorrang vor diesen Einstellungen haben. Zum Beispiel kann der Abschnitt `[Adobe Reader 8.exe]` in der `Package.ini`-Datei für eine Adobe Reader-Anwendung Einstellungen enthalten, die Vorrang gegenüber den allgemeineren `[BuildOptions]`-Parametern haben. Die anwendungsspezifischen Parameter weisen die Einstiegspunkte für die Anwendungen aus, die Sie während des Build-Prozesses erstellen.

Die Parameter `[FileList]`, `[Compression]` und `[Isolation]` fungieren als `[BuildOptions]`-Parameter, sind jedoch aus Gründen der Abwärtskompatibilität gesondert gruppiert. Sie können die Überschrift `[FileList]` manuell zur Datei hinzufügen, wenn Sie den Parameter `ExcludePattern` hinzufügen.

Parameter, die nicht zu den Standardabschnitten gehören, können unter einer beliebigen Überschrift angegeben werden. Parameter müssen nicht in alphabetischer Reihenfolge angeordnet sein.

Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden

Sie können bestimmte Parameter auf die Datei `Package.ini` oder die Datei `##Attributes.ini` anwenden, sofern es erforderlich ist, die Einstellungen in `Package.ini` auf Verzeichnisebene zu überschreiben.

Sie können die Parameter `DirectoryIsolationMode`, `CompressionType` und `ExcludePattern` in der Datei `##Attributes.ini` verwenden. Die `##Attributes.ini`-Datei ist in den Ordnermakros des Projektordners vorhanden.

Weitere Informationen über die `##Attributes.ini`-Datei erhalten Sie unter „Ändern der Einstellungen in der ##Attributes.ini-Datei“ auf Seite 25.

Konfigurieren der ThinApp-Laufzeit

Sie können ThinApp-Parameter für Aufgaben zur Laufzeitkonfiguration ändern, die sich auf die Leistung beim Anwendungsstart und auf virtuelle Computernamen auswirken.

NetRelaunch

Der Parameter `NetRelaunch` legt fest, ob eine Anwendung, die von einer Netzwerkfreigabe oder einem Wechseldatenträger aus ausgeführt wird, stattdessen von der lokalen Festplatte aus erneut gestartet werden soll, damit der Start von Anwendungen schneller abläuft.

ThinApp legt einen Anfangswert für den Parameter `NetRelaunch` fest, der erkennt, ob eine Anwendung von einem Netzlaufwerk oder von einem Wechseldatenträger aus ausgeführt wird, und verwendet eine ausführbare Stub-Datei, um einen Neustart der Anwendung von der lokalen Festplatte aus durchzuführen. Dieser Prozess löst Leistungsprobleme, die entstehen, wenn Symantec AntiVirus versucht, eine vollständige Überprüfung für ausführbare Dateien, die von einer Netzwerkfreigabe oder von einem Wechseldatenträger gestartet werden, sowie für ausführbare Dateien, die die anfänglichen Netzwerkverbindungen herstellen, durchzuführen. Die Überprüfung kann sich auf die Startdauer von großen ausführbaren Dateien auswirken.

Da zahlreiche Desktop-Computer mit Symantec AntiVirus ausgestattet sind, ermöglicht ThinApp, dass die Anwendungen von einer Netzwerkfreigabe aus gestartet werden, ohne dass Verzögerungen durch die Überprüfung entstehen. Wird die Anwendung von einer Netzwerkfreigabe oder von einem Wechseldatenträger aus ausgeführt, erstellt ThinApp eine ausführbare Stub-Datei in dem Verzeichnis, das der Parameter `CachePath` auf der lokalen Festplatte festlegt und startet die Anwendung von dieser ausführbaren Stub-Datei aus neu. Die ausführbare Stub-Datei kann die Laufzeit vom großen Paket laden und den Rest der Anwendung von ihrem ursprünglichen Netzwerkspeicherort lesen. Symantec AntiVirus sieht die Anwendung als lokal an, und überprüft die größere ausführbare Datei der Netzwerkfreigabe oder auf dem Wechseldatenträger nicht.

Beispiele

Wenn Ihre Anwendung klein ist oder wenn Sie sicher sind, dass Symantec AntiVirus nicht auf den Desktop-Computern installiert ist, auf denen die Anwendung ausgeführt werden soll, können Sie den Parameter `NetRelaunch` entsprechend ändern, um eine bessere Leistung beim ersten Starten zu erzielen.

```
[BuildOptions]
NetRelaunch=0
```

RuntimeEULA

Der Parameter `RuntimeEULA` steuert die Anzeige der Endbenutzerlizenzvereinbarung (EULA) für das Paket. Dieser Parameter bezieht sich auf ältere EULA-Anforderungen. VMware erfordert keine Laufzeit-EULA für ThinApp-Pakete.

Der Wert dieses Parameters darf nicht geändert werden.

Beispiele

Der Parameter `RuntimeEULA` verhindert die Anzeige der Endbenutzerlizenzvereinbarung.

```
[BuildOptions]
;Default: do not show an Eula (;Standard: Endbenutzerlizenzvereinbarung nicht anzeigen)
RuntimeEULA=0
```

VirtualComputerName

Der Parameter `VirtualComputerName` legt fest, ob der Computername virtualisiert werden soll, um Namenskonflikte zwischen dem Bereitstellungssystem und dem Kapselungssystem zu vermeiden.

Anwendungen können den Namen des Computers, auf dem sie installiert wurden oder über den sie auf eine Datenbank zugreifen, und den Namen des Computers in der Verbindungszeichenfolge verwenden. Da für Kapselung und Bereitstellung unterschiedliche Systeme verwendet werden, müssen gekapselte Anwendungen, die einen Computernamen benötigen, diesen Computernamen virtualisieren, um sicherzustellen, dass die Anwendung auf einem beliebigen Rechner ausgeführt werden kann.

ThinApp kommentiert die ursprüngliche Einstellung des Parameters `VirtualComputerName` aus. Dieser Parameter verwendet eine Zeichenfolge, die von den API-Funktionen `GetComputerName` und `GetComputerNameEx` in einer virtuellen Anwendung ausgegeben wird.

Beispiele

Trägt das System, mit dem die Kapselung ausgeführt wurde, nicht den Namen LOCALHOST, kommentiert ThinApp den Parameter `VirtualComputerName` aus.

```
;VirtualComputerName=<Originaler_Maschinenname>
```

Wenn Sie eine neu aufgesetzte Maschine in `LOCALHOST` umbenennen, bevor Sie den Kapselungsprozess ausführen, aktiviert die `Package.ini`-Datei den Eintrag `VirtualComputerName`. Die virtuelle Anwendung verwendet den Namen `LOCALHOST`, weil jeder Computer, auf dem die Anwendung ausgeführt wird, diesen Wert als Computernamen erhält.

```
VirtualComputerName=LOCALHOST
```

Wenn Sie den Befehl `GetComputerName` oder `GetComputerNameEx` eingeben, gibt der Rechner den Wert `LOCALHOST` zurück. Wenn das Windows-System die API-Funktionen `GetComputerName` und `GetComputerNameEx` benötigt, um standardmäßig zu funktionieren und den tatsächlichen Namen auszugeben, auf dem die Anwendung ausgeführt wird, sollte der Rechner nicht in `LOCALHOST` umbenannt werden.

Abgesehen von der Angabe einer literalen Zeichenfolge wie LOCALHOST können Sie auch eine Umgebungsvariable angeben.

```
VirtualComputerName=%VCOMPNAME%
```

Wenn Sie eine Umgebungsvariable angeben, wird als Wert der Wert der Umgebungsvariablen ausgegeben. Lautet der Wert des Parameters `VirtualComputerName` %VCOMPNAME%, und ist als Umgebungsvariable für %VCOMPNAME% der Wert `EnvCompName` angegeben, so gibt die API-Funktion `GetComputerName` den Wert `EnvCompName` aus.

Wow64

Der Parameter `Wow64` simuliert eine 32-Bit-Umgebung für 32-Bit-Anwendungen, die nicht auf einem 64-Bit-Windows-Betriebssystem ausgeführt werden können.

Versucht eine 32-Bit-Anwendung, eine eigene 64-Bit-Registrierungsumleitung zu verarbeiten, so können Sie diesen Parameter vor der Erstellung eines Projekts aktivieren. ThinApp kommentiert den Parameter aus, um die Emulation von Windows auf Windows 64-Bit (WOW64) zu verhindern.

Beispiele

Sie können die Auskommentierung des Parameters `Wow64` aufheben, um auf einem 64-Bit-Betriebssystem eine 32-Bit-Umgebung für 32-Bit-Anwendungen zu simulieren. Beispielsweise funktioniert eine virtualisierte 32-Bit-Oracle-Anwendung möglicherweise nicht auf einem 64-Bit-Betriebssystem.

```
[BuildOptions]
Wow64=0
```

QualityReportingEnabled

Der Parameter `QualityReportingEnabled` gibt an, ob VMware anonyme Daten über ein Paket erfassen kann, um die ThinApp-Anwendungsunterstützung zu verbessern. VMware erfasst Anwendungsinformationen, wie z. B. die Version und die Anzahl der Anwendungsfehler.

Wenn Sie der Erfassung anonymer Daten während des Kapselungsprozesses zustimmen, lädt der Parameter `QualityReportingEnabled` alle 10 Tage Daten hoch.

Beispiele

Sie können die Datenerfassung durch ThinApp deaktivieren, indem Sie die Einstellung des Parameters `QualityReportingEnabled` entsprechend ändern.

```
[BuildOptions]
QualityReportingEnabled=0
```

Konfigurieren des Isolationsmodus

Sie können die ThinApp-Parameter ändern, die den Lese- und Schreibzugriff auf das Dateisystem und die Registrierungsschlüssel bestimmen.

DirectoryIsolationMode

Der Parameter `DirectoryIsolationMode` gibt den Grad des Lese- und Schreibzugriffs für Verzeichnisse auf das physische Dateisystem an.

Beim Kapselungsprozess wird der Anfangswert des Parameters `DirectoryIsolationMode` in der Datei `Package.ini` festgelegt. Dieser Parameter steuert den Standard-Isolationsmodus für Dateien, die von der virtuellen Anwendung erstellt wurden, sofern Sie nicht einen anderen Isolationsmodus für ein einzelnes Verzeichnis in der `##Attributes.ini`-Datei angeben. Alle nicht angegebenen Verzeichnisse, wie zum Beispiel `C:\myfolder`, übernehmen den Isolationsmodus von der `Package.ini`-Datei.

ThinApp stellt während des Kapselungsprozesses nur die Isolationsmodusoptionen „Merged“ und „WriteCopy“ zur Verfügung. Sie können den Isolationsmodus „Full“ außerhalb des Assistenten zur Sicherung der virtuellen Umgebung verwenden.

Mit dem Isolationsmodus „Merged“ können Anwendungen Elemente im physischen Dateisystem außerhalb des virtuellen Anwendungspakets lesen und ändern. Für einige Anwendungen ist es erforderlich, DLL-Dateien und Registrierungsdaten im lokalen Systemabbild zu lesen. Der Vorteil des Modus „Merged“ liegt darin, dass von Benutzern gespeicherte Dokumente im physischen System an dem vom Benutzer erwarteten Speicherort anstatt in der Sandbox angezeigt werden. Der Nachteil ist, dass dieser Modus das Systemabbild überhäufen könnte. Ein Beispiel für diese Überhäufung können Markierungen für die Erstausführung von Shareware-Anwendungen an zufälligen Computerspeicherorten als Teil des Lizenzierungsvorgangs sein.

Mit dem Isolationsmodus „WriteCopy“ kann ThinApp Schreibvorgänge abfangen und in die Sandbox umleiten. Sie können den Isolationsmodus „WriteCopy“ für ältere oder nicht vertrauenswürdige Anwendungen verwenden. Obwohl das Auffinden von Benutzerdatendateien, die sich in der Sandbox statt im physischen System befinden, durch diesen Modus möglicherweise erschwert wird, ist dieser Modus für gesperrte Desktops nützlich, wenn Sie Benutzer daran hindern möchten, das lokale Dateisystem zu beeinflussen.

Beim Isolationsmodus „Full“ blockiert ThinApp die Sichtbarkeit von Systemelementen außerhalb des virtuellen Anwendungspakets. Dieser Modus beschränkt Änderungen an Dateien oder Registrierungsschlüsseln auf die Sandbox und stellt sicher, dass keine Interaktion mit der Umgebung außerhalb des virtuellen Anwendungspakets stattfindet. Der Isolationsmodus „Full“ verhindert Anwendungskonflikte zwischen der virtuellen Anwendung und den auf dem physischen System installierten Anwendungen. Verwenden Sie den Isolationsmodus Full (Voll) nicht in der `Package.ini`-Datei, da der Modus die Fähigkeit System-DLLs zu erkennen und zu laden, blockiert. Sie können den Isolationsmodus Full (Voll) als Überschreibmechanismus in den `##Attributes.ini`-Dateien verwenden.

ThinApp speichert die Isolationsmodi für die Registrierung und das Dateisystem während der Laufzeit in der Sandbox. Wenn Sie den Isolationsmodus für das Projekt ändern und die ausführbare Datei erneut erstellen, müssen Sie möglicherweise die Sandbox löschen, bevor die Änderung wirksam wird.

Informationen zu den Definitionen und dem Effekt der Isolationsmodi erhalten Sie unter „Definition von Isolationsmodi für das physische Dateisystem“ auf Seite 18.

Beispiele

Bei aktiviertem Isolationsmodus „WriteCopy“ können Sie den Parameter `DirectoryIsolationMode` ändern, um sicherzustellen, dass die Anwendung zwar Ressourcen auf dem lokalen Rechner lesen, jedoch keine Schreibvorgänge auf dem Hostcomputer vornehmen kann. Dies ist die Standardeinstellung für das Dienstprogramm `snapshot.exe`. Dieser Parameter muss unter der Überschrift `[Isolation]` angegeben werden.

```
[Isolation]
DirectoryIsolationMode=WriteCopy
```

Sie können den Isolationsmodus „Merged“ zuweisen, um sicherzustellen, dass die Anwendung Ressourcen an einem beliebigen Speicherort auf dem Computer lesen und an einen beliebigen Speicherort auf dem Computer schreiben kann, sofern das Paket nichts Gegenteiliges angibt. Dies ist die Standardeinstellung für den Setup Capture-Assistenten.

```
[Isolation]
DirectoryIsolationMode=Merged
```

RegistryIsolationMode

Der Parameter `RegistryIsolationMode` steuert den Isolationsmodus für Registrierungsschlüssel im Paket. Diese Einstellung gilt für die Registrierungsschlüssel, die keine expliziten Einstellungen aufweisen.

Bei diesem Kapselungsprozess wird der Wert dieses Parameters nicht festgelegt. Sie können den Isolationsmodus für Registrierungsschlüssel nur in der Datei `Package.ini` konfigurieren. ThinApp legt als anfänglichen Isolationsmodus für Registrierungsschlüssel „WriteCopy“ fest. Informationen über Optionen für Isolationsmodi erhalten Sie unter „DirectoryIsolationMode“ auf Seite 60.

Verwenden Sie den Isolationsmodus „Full“ nicht in der Datei `Package.ini`, da der Modus die Fähigkeit System-DLLs zu erkennen und zu laden, blockiert. Sie können den Isolationsmodus „Full“ als Überschreibmechanismus verwenden. Ausnahmen von der Konfiguration des Parameters `RegistryIsolationMode` können Sie im Projektverzeichnis in den Textdateien für Registrierungsschlüssel angeben. Eine Ausnahme kann in einer dieser Dateien, z. B. `HKEY_CURRENT_USER.txt`, wie folgt aussehen: `isolation_full HKEY_CURRENT_USER\Software\Macromedia`.

Alle Laufzeitänderungen an virtuellen Dateien in der gekapselten Anwendung sind in der Sandbox gespeichert, unabhängig von der Einstellung des Isolationsmodus. Während der Laufzeit sind virtuelle und physische Registrierungsdateien für eine Anwendung nicht unterscheidbar. Virtuelle Registrierungsdateien haben allerdings immer Vorrang vor physischen, wenn beide am selben Speicherort vorhanden sind. Wenn virtuelle und physische Einträge am selben Speicherort vorhanden sind, beeinflussen Isolationsmodi den Zugriff auf diese Einträge nicht, weil die Anwendung immer mit virtuellen Elementen interagiert.

Wenn Updates externer Gruppenrichtlinien getrennt vom Paket durch die physische Registrierung auftreten, müssen Sie möglicherweise virtuelle Registrierungsdateien aus einem Paket entfernen und sich vergewissern, dass die übergeordnete Datei dieser virtuellen Registrierungsdateien nicht den Isolationsmodus „Full“ verwendet. Da untergeordnete Dateien Isolationsmodi von übergeordneten Dateien übernehmen, kann der Isolationsmodus „Full“ in einer übergeordneten Datei die Sichtbarkeit von physischen untergeordneten Dateien für eine Anwendung blockieren.

Beispiele

Sie können den Parameter `RegistryIsolationMode` ändern, sodass sichergestellt wird, dass die Anwendung Schlüssel vom Hostcomputer lesen, jedoch nicht auf den Hostcomputer schreiben kann.

```
[Isolation]
RegistryIsolationMode=WriteCopy
```

Sie können sicherstellen, dass die Anwendung in alle Schlüssel auf dem Computer schreiben kann, außer diejenigen, für die das Paket etwas Gegenteiliges angibt.

```
[Isolation]
RegistryIsolationMode=Merged
```

Konfigurieren von Datei- und Protokollzuordnungen

Sie können mithilfe der ThinApp-Parameter festlegen, dass den Anwendungen Dateierweiterungen zugeordnet und dass Protokolle festgelegt werden, die für die physische Umgebung sichtbar sind.

FileTypes

Der Parameter `FileTypes` listet Dateierweiterungen auf, die vom Dienstprogramm `thinreg.exe` mit einer ausführbaren Datei verknüpft werden.

Beim Kapselungsprozess werden Anfangswerte generiert, denen Sie keine weiteren hinzufügen können. Sie können Dateierweiterungen entfernen, die nicht mit dem virtuellen Paket verknüpft werden sollen. Geben Sie keine Trennzeichen zwischen den Dateierweiterungen in der Liste an.

Beispiele

Für ein Paket von Microsoft Word 2007 werden `.doc.docx` als Werte des Parameters `FileTypes` angegeben. Wenn Sie Microsoft Office 2007 kapseln und Microsoft Office 2003 in der physischen Umgebung installiert ist, können Sie die Dateierweiterung `.doc` aus dem Parameter `FileTypes` entfernen und die Dateierweiterung `.docx` im Parameter beibehalten, um sicherzustellen, dass Dateien mit der Erweiterung `.doc` von Microsoft Word 2003 und Dateien mit der Erweiterung `.docx` von Microsoft Word 2007 geöffnet werden.

```
[Microsoft Office Word 2007.exe]
FileTypes=.docx
```

Der Kapselungsprozess kann Dateitypzuordnungen für die Dateierweiterungen `.doc` und `.dot` erstellen und diese mit Microsoft Word verknüpfen.

```
[Microsoft Office Word 2003.exe]
ReadOnlyData=bin\Package.ro.tvr
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\WINWORD.EXE
FileTypes=.doc.dot
```

Protocols

Der Parameter `Protocols` gibt die Protokolle an, die für Anwendungen in der physischen Umgebung sichtbar sind, zum Beispiel HTTP. Dieser Parameter ist dem Parameter `FileTypes` ähnlich, bezieht sich aber auf Anwendungen, die Protokolle statt Dateitypen behandeln.

Beim Kapselungsprozess werden Anfangswerte generiert, denen Sie keine weiteren hinzufügen können. Für Browser oder andere Anwendungen können Sie Einträge entfernen.

Beispiele

Mit dem Kapselungsprozess können Protokolle, wie das Protokoll `mailto` für ein Microsoft Outlook-Paket, im Parameter `Protocols` angegeben werden.

```
[Microsoft Office Outlook 2007.exe]
Protocols=feed;feeds;mailto;Outlook.URL.mailto;stssync;webcal;webcals
```

Konfigurieren der Build-Ausgabe

Sie können mithilfe der ThinApp-Parameter festlegen, dass der Speicherort der Build-Ausgabe und der Dateien im Paket angegeben wird.

ExcludePattern

Der Parameter `ExcludePattern` schließt Dateien bzw. Verzeichnisse während des Build-Prozesses für die Anwendung aus. Sie müssen vor diesem Parametereintrag die Überschrift `[FileList]` hinzufügen.

Sie können ein Komma verwenden, um die Muster in der Liste zu trennen. Platzhalter (*)können mit keinem der Zeichen oder mindestens mit einem der Zeichen übereinstimmen, und Fragezeichen (?) stimmen mit genau einem Zeichen überein. Die Syntax ist vergleichbar mit dem `dir`-Befehl in DOS; jedoch können Sie Platzhalterzeichen auf Verzeichnis- und Dateinamen anwenden.

Sie können den Parameter `ExcludePattern` in der `Package.ini`-Datei angeben. Der Musterausschluss wird dann auf die gesamte Verzeichnisstruktur angewandt. Sie können den Parameter auch in der `##Attributes.ini`-Datei angeben. In diesem Fall fügt ThinApp den Musterausschluss in die aktuelle Liste mit Ausschlüssen wendet, aber die Einstellungen nur auf das spezifische Verzeichnis und dessen Unterverzeichnisse an. Sie können unterschiedliche Ausschlusslisten für verschiedene Verzeichnisse in Ihrem Projekt erstellen.

Beispiele

Wenn Sie Pakete in einem Versionskontrollsystem speichern und Sie die Versionskontrollinformationen aus dem virtuellen Dateisystem ausschließen möchten, können Sie alle mit `.svn` oder `.cvs` bezeichneten Verzeichnisse und alle Unterverzeichnisse ausschließen.

```
[FileList]
ExcludePattern=\.svn,\.cvs
```

Das Muster stimmt nicht mit Dateinamen oder Verzeichnissen überein, die `.svn` oder `.cvs` in der Mitte der Zeichenfolge enthalten.

Sie können alle Pfade ausschließen, die auf `.bak` oder `.msi` enden.

```
[FileList]
ExcludePattern=*.bak,*.msi
```

Icon

Der Parameter `Icon` gibt die Symboldatei an, die der generierten ausführbaren Datei zugeordnet werden soll. Dieses Symbol wird in der Anwendung, z. B. Microsoft Word, und in den der Anwendung zugeordneten Dateien, z. B. Dateien mit der Dateierweiterung `.doc`, angezeigt.

Jeder Anwendung ist ein eigenes Symbol zugeordnet, das in einer Datei mit der Erweiterung `.ico`, innerhalb der `.exe`-Datei der Anwendung oder innerhalb einer `.dll`-Datei, gespeichert ist. Beim Kapselungsprozess werden die Symbole den ausführbaren Dateien zugeordnet. Die Anwendung verwendet das Hauptgruppensymbol der ausführbaren Datei, die im Parameter `Source` angegeben ist, und die spezifische Symbolressource, auf die das Gruppensymbol verweist.

Beispiele

Sie können mit dem Parameter `Icon` festlegen, dass ein alternatives Symbol verwendet werden soll, indem Sie eine andere ausführbare Datei als die im Parameter `Source` angegebene Datei festlegen. Solche alternativen Symbole können für Drittanbieter nützlich sein.

```
[<Eigene_Anw>.exe]
Source=%ProgramFilesDir%\<Eigene_Anw>\app.exe
Icon=%ProgramFilesDir%\<Eigene_Anw>\app2.exe
```

Durch Anhängen von `,1 ,2` am Ende des Symbolpfades können Sie einen Symbolsatz angeben.

```
[<Eigene_Anw>.exe]
Source=%ProgramFilesDir%\<Meine_Anw>\<Anw>.exe
Icon=%ProgramFilesDir%\<Eigene_Anw>\<app2>.exe,1
```

Mithilfe einer `.ico`-Datei können Sie das Anwendungssymbol angeben.

```
[<Eigene_Anw>.exe]
Source=%ProgramFilesDir%\<Eigene_Anw>\<app>.exe
Icon=%ProgramFilesDir%\<Eigene_Anw>\<Eigenes_Symbol>.ico
```

OutDir

Der Parameter `OutDir` gibt das Verzeichnis an, in dem die `build.bat`-Ausgabe gespeichert wird.

Der Wert dieses Parameters darf nicht geändert werden.

Beispiele

Der statische Wert des Parameters `OutDir` gibt das `bin`-Verzeichnis des Projekts an.

```
[BuildOptions]
OutDir=bin
```

RetainAllIcons

Der Parameter `RetainAllIcons` listet alle ursprünglichen Symbole der ausführbaren Datei auf, die im Parameter `Source` in der Anwendung angegeben ist.

Die Symbole, die keiner ausführbaren Datei einer Anwendung zugeordnet sind, befinden sich im virtuellen Dateisystem des Pakets. Mit dem Parameter `RetainAllIcons` wird festgelegt, ob die nicht verwendeten Symbole aus dem virtuellen Dateisystem in die ausführbare Datei kopiert werden sollen. Um möglichst wenig Speicherplatz zu belegen, legt ThinApp einen Anfangswert fest, mit dem nicht verwendete Symbole aus dem für die physische Umgebung sichtbaren Teil der ausführbaren Datei entfernt werden.

Beispiele

Wenn alle ursprünglichen Symbole der Anwendung gespeichert bleiben sollen, können Sie den Parameter `RetainAllIcons` entsprechend ändern.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
RetainAllIcons=1
```

Konfigurieren von Berechtigungen

Sie können die ThinApp-Parameter für Sicherheitsaufgaben, die den Benutzerzugriff auf Pakete definieren, anpassen und den DEP-Schutz (Data Execution Prevention) ändern.

AccessDeniedMsg

Der Parameter `AccessDeniedMsg` enthält eine Fehlermeldung, die für Benutzer angezeigt wird, die zur Ausführung eines Pakets nicht berechtigt sind.

ThinApp legt anfänglich eine Meldung fest, mit der der Benutzer zur Kontaktaufnahme mit dem Administrator aufgefordert wird.

Beispiele

Sie können dem Parameter `AccessDeniedMsg` die Telefonnummer des technischen Supports hinzufügen.

```
[BuildOptions]
PermittedGroups=Administrator;Office-Benutzer
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1-800-822-2992
```

AddPageExecutePermission

Der Parameter `AddPageExecutePermission` unterstützt Anwendungen, die in einer DEP-Umgebung (Data Execution Prevention) nicht funktionieren.

Die DEP-Funktion von Windows XP SP2, Windows Server 2003 und höheren Versionen des Betriebssystems schützt vor bestimmten Sicherheitslücken, die bei Pufferüberlauf auftreten. Diese Funktion schafft Kompatibilitätsprobleme. Die Funktion ist unter Windows XP SP2 standardmäßig deaktiviert, und Sie können eine rechnerspezifische Auswahlliste mit den Anwendungen verwenden, auf die der DEP-Schutz angewandt werden soll. Die Auswahlregeln können schwierig zu verwalten sein, wenn eine große Anzahl an Rechnern und Anwendungen beteiligt ist. Der Parameter `AddPageExecutePermission` weist ThinApp an, bestimmten Seiten, die von einer Anwendung zugewiesen werden, eine Ausführungsberechtigung hinzuzufügen. Die Anwendung kann ohne vorherige Änderung der Auswahlliste auf Rechnern ausgeführt werden, bei denen der DEP-Schutz aktiviert ist.

ThinApp legt für den Parameter `AddPageExecutePermission` einen Anfangswert fest, der jegliche Änderungen am DEP-Schutz verhindert.

Beispiele

Sie können den Parameter `AddPageExecutePermission` so ändern, dass bestimmten Seiten, die von einer Anwendung zugewiesen werden, eine Ausführungsberechtigung hinzugefügt wird. ThinApp führt einen Code von Speicherseiten aus, die die Anwendung angibt. Dies ist nützlich für Anwendungen, die das Programm und seine Daten in einem Bereich des Arbeitsspeichers kombinieren.

```
[BuildOptions]
;Disable some Data Execution protections for this particular application (;Deaktivierung einiger
               DEP-Schutzmaßnahmen für diese spezielle Anwendung)
AddPageExecutionPermission=1
```

PermittedGroups

Der Parameter `PermittedGroups` schränkt die Verwendung eines Pakets auf eine bestimmte Gruppe von Active Directory-Benutzern ein.

Sie können Gruppennamen, SID-Zeichenfolgen oder eine Mischung aus Gruppennamen und SID-Zeichenfolgen in einer Zeile mit dem Parameter `PermittedGroups` angeben. Wenn Sie einen domänenbasierten Gruppennamen verwenden, müssen Sie eine Verbindung zur betreffenden Domäne herstellen, wenn Sie das Anwendungspaket erstellen. Wenn Sie dem Parameterwert eine SID hinzufügen, brauchen Sie keine Verbindung zu der Domäne herzustellen, in der die SID definiert ist.

Mithilfe der Dienste für die Active Directory-Domäne werden Sicherheitsgruppen und Verteilungsgruppen erstellt. Dieser Parameter kann nur verschachtelte Sicherheitsgruppen unterstützen. Wenn ein Benutzer beispielsweise Mitglied der Sicherheitsgruppe A ist und wenn diese Gruppe A Mitglied der Sicherheitsgruppe B ist, dann kann ThinApp den Benutzer als Mitglied der Gruppe A und der Gruppe B erkennen.

Wenn ThinApp eine Anwendung erstellt, geht ThinApp davon aus, dass alle angegebenen Gruppennamen gültig sind, und konvertiert die Namen in SID-Werte. ThinApp kann die Gruppenzugehörigkeit während der Laufzeit mithilfe von im Cache gespeicherten Benutzerdaten auflösen. Sie können Laptop-Benutzer weiterhin authentifizieren, selbst wenn diese nicht mit dem Netzwerk verbunden sind. Hat der Benutzer keine Berechtigung zur Ausführung des Pakets, so können Sie den Parameter `AccessDeniedMsg` derart anpassen, dass der Benutzer einen entsprechenden Hinweis erhält.

Sie können den Parameter `PermittedGroups` entweder unter der Überschrift `[BuildOptions]` angeben, damit er sich auf das Paket auswirkt, oder unter der Überschrift `[<application>.exe]`, damit er sich nur auf eine bestimmte Anwendung auswirkt. Der Wert `[<Anwendung>.exe]` überschreibt den Standardwert `[BuildOptions]` dieser bestimmten Anwendung.

Beispiele

Sie können im Parameter `PermittedGroups` eine Liste mit durch Semikolons getrennten Active Directory-Benutzergruppennamen angeben. Die Parameter im Abschnitt `[BuildOptions]` legen die globalen Einstellungen für das gesamte Projekt fest.

```
[BuildOptions]
PermittedGroups=Administrator;Office-Benutzer
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1-800-822-2992.
```

Sie können eine Benutzergruppeneinstellung für eine bestimmte Anwendung angeben, die die globale Einstellung `PermittedGroups` überschreibt.

```
[App1.exe]
PermittedGroups=Gast
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1-800-822-2992.
```

Wenn Sie keine `PermittedGroups`-Einstellung für eine Anwendung angeben, übernimmt die Anwendung den globalen Wert `PermittedGroups` in der Sektion `[BuildOptions]`.

```
[App2.exe]
...
```

Sie können in einem Eintrag für den `PermittedGroups`-Parameter sowohl Gruppennamen als auch SID-Zeichenfolgen verwenden.

```
PermittedGroups=S-1-5-32-544;Office-Benutzer
```

UACRequestedPrivilegesLevel

Der Parameter `UACRequestedPrivilegesLevel` gibt Berechtigungen für Programme an, für die UAC-Informationen erforderlich sind. Dieser Parameter wirkt sich auf Benutzer aus, die mit Windows Vista oder höheren Betriebssystemversionen arbeiten.

Sie können mithilfe der folgenden Werte Berechtigungen angeben:

- asInvoker

  Dieser Wert verwendet das Profil in Vista.

- requireAdministrator
- highestAvailable

  Dieser Wert verwendet die höchste verfügbare Berechtigung, mit der die UAC-Eingabeaufforderung verhindert werden kann.

Wenn Sie keine Berechtigungen angeben, ordnet ThinApp keinen Standardwert zu, sondern arbeitet gemäß der asInvoker-Einstellung.

Beispiele

Sie können mit dem Parameter `UACRequestedPrivilegesLevel` Administratorberechtigungen für ein Programm angeben.

```
[BuildOptions]
UACRequestedPrivilegesLevel=requireAdministrator
```

UACRequestedPrivilegesUIAccess

Der Parameter `UACRequestedPrivilegesUIAccess` steuert den Zugriff auf die Benutzeroberfläche von Windows Vista oder höheren Betriebssystemversionen. Auf diese Weise werden einige Elemente der Benutzeroberfläche durch die Betriebssysteme geschützt.

ThinApp weist dem Parameter `UACRequestedPrivilegesUIAccess` einen Anfangswert zu, mit dem der Zugriff durch Anwendungen auf geschützte Elemente blockiert wird. Obwohl Sie dem Parameter `UACRequestedPrivilegesUIAccess` die Werte `true` oder `false` zuweisen können, um den Zugriff auf die Benutzeroberfläche zu steuern, dient der Parameter eigentlich zur Unterstützung von Microsoft-Einstellungen.

Beispiele

Sie können den Anfangswert des Parameters `UACRequestedPrivilegesUIAccess` beibehalten, um sicherzustellen, dass die virtuelle Anwendung keinen Zugriff auf geschützte Elemente hat.

```
[BuildOptions]
UACRequestedPrivilegesUiAccess=false
```

Konfigurieren von Objekten und DLL-Dateien

Durch Änderung der ThinApp-Parameter können Sie den COM-Objektzugriff und die Anforderungen zum Laden von DLLs angeben.

ExternalCOMObjects

Der Parameter `ExternalCOMObjects` legt fest, ob Windows COM-Objekte in der physischen Umgebung anstatt in der virtuellen Umgebung erstellt und ausführt, um die Anwendungskompatibilität mit ThinApp zu ermöglichen. Außerhalb der virtuellen Umgebung erstellte COM-Objekte werden immer in der physischen Umgebung ausgeführt.

Der von ThinApp für den Parameter `ExternalCOMObjects` festgelegte Anfangswert gibt an, dass die COM-Objekte in der virtuellen Umgebung erstellt und ausgeführt werden.

COM unterstützt ausführbare Server außerhalb von Prozessen und dienstbasierte COM-Objekte. Wenn eine Anwendung COM-Objekte erstellen kann, die Modifizierungen auf dem Hostcomputer generieren, ist die Integrität des Hostcomputers in Gefahr. Wenn ThinApp außerhalb der Prozesse und dienstbasierten COM-Objekte in der virtuellen Umgebung ausgeführt wird, speichert ThinApp alle von den COM-Objekten durchgeführten Änderungen in der Sandbox.

Dieser Parameter wird nicht durch den Kapselungsprozess generiert. Sie können diesen Parameter zu der Datei `Package.ini` hinzufügen.

Beispiele

Wenn Sie die Fehlerbehebung mit VMware-Unterstützung ausführen und dabei feststellen, dass von einer Anwendung COM-Objekte implementiert werden, die nicht mit ThinApp kompatibel sind, können Sie den Parameter `ExternalCOMObjects` so anpassen, dass die COM-Objekte außerhalb der virtuellen Umgebung ausgeführt werden. Sie können die CLSID-Schlüssel auflisten.

```
[BuildOptions]
ExternalCOMObjects={8BC3F05E-D86B-11D0-A075-00C04FB68820};{7D096C5F-AC08-4F1F-BEB7-5C22C517CE39}
```

ExternalDLLs

Der Parameter `ExternalDLLs` kann erzwingen, dass Windows bestimmte DLL-Dateien aus dem virtuellen Dateisystem lädt.

ThinApp legt einen Anfangswert fest, mit dem DLL-Dateien aus dem virtuellen Dateisystem geladen werden und der Ladeprozess für DLL-Dateien im physischen Dateisystem an Windows übergeben wird. Unter bestimmten Umständen muss Windows eine DLL-Datei im virtuellen Dateisystem laden. Möglicherweise haben Sie eine DLL-Datei, die sich mithilfe von Windows-Hooks in andere Prozesse einfügt. Die DLL-Datei, die den Haken implementiert, muss auf dem Host-Dateisystem verfügbar sein, und Windows muss die Datei laden. Wenn Sie eine DLL-Datei im Parameter `ExternalDLLs` angeben, extrahiert ThinApp die Datei aus dem virtuellen Dateisystem in die Sandbox und weist Windows an, die Datei zu laden.

Virtuelle Diktiersoftware ist eine Software, die über eine Schnittstelle eine Verbindung zu nativen Anwendungen herstellen kann, die Informationen zwischen DLL-Dateien übergeben. ThinApp kann das Laden von DLL-Dateien in die virtuelle Umgebung an Windows übergeben, um sicherzustellen, dass über eine Schnittstelle eine Verbindung zwischen lokalen Anwendungen und den DLL-Dateien hergestellt werden kann.

Der Parameter `ExternalDLLs` unterstützt keine DLL-Dateien, die innerhalb des virtuellen Dateisystems von anderen DLL-Dateien abhängig sind. In diesem Fall kann Windows die DLL-Datei nicht laden.

Beispiele

Sie können mit dem Parameter `ExternalDLLs` erzwingen, dass Windows die Dateien `inject.dll` und `injectme2.dll` aus dem virtuellen Dateisystem lädt.

```
[BuildOptions]
ExternalDLLs=inject.dll;injectme2.dll
```

ForcedVirtualLoadPaths

Der Parameter `ForcedVirtualLoadPaths` weist ThinApp an, DLL-Dateien als virtuelle DLL-Dateien zu laden, selbst wenn sich die Dateien außerhalb des Pakets befinden. Dieser Parameter ist nützlich, wenn die Anwendung externe DLL-Systemdateien laden muss, die von DLL-Dateien innerhalb des Pakets abhängig sind.

Die DLL-Pfade können Makros enthalten. Verwenden Sie zum Trennen mehrerer Pfade Semikolons.

Mit diesem Parameter wird dasselbe Ergebnis erzielt wie mit der API-Funktion `AddForcedVirtualLoadPath`. Siehe „AddForcedVirtualLoadPath“ auf Seite 115.

Beispiele

Sie können den Parameter `ForcedVirtualLoadPaths` ändern, wenn Sie eine von externen DLL-Dateien abhängige Anwendung haben. Wenn Sie Microsoft Office ohne Microsoft Outlook kapseln und auf dem lokalen System eine native Version von Microsoft Outlook vorhanden ist, können Sie keine E-Mail von der virtuellen Version von Microsoft Excel aus senden, weil die native Datei `envelope.dll`, die zusammen mit Microsoft Outlook installiert wurde, von der Datei `mso.dll`, die ThinApp in die virtuelle Umgebung lädt, abhängig ist. Sie können jedoch erzwingen, dass ThinApp die Datei `envelope.dll` in die virtuelle Umgebung anstatt in die native Umgebung lädt.

```
[BuildOptions]
ForcedVirtualLoadPaths=%ProgramFilesDir%\Microsoft Office\Office10\envelope.dll
```

IsolatedMemoryObjects

Der Parameter `IsolatedMemoryObjects` listet die freigegebenen Speicherobjekte auf, die von anderen Anwendungen oder von Systemobjekten isoliert werden sollen.

Anwendungen, die die Windows-Funktionen `CreateFileMapping` und `OpenFileMapping` verwenden, erstellen gemeinsam genutzte Arbeitsspeicherobjekte. Wenn Sie Speicherobjekte nicht isolieren, kann es zu Konflikten zwischen den virtuellen Anwendungen und den nativen Anwendungen kommen, von denen diese Objekte freigegeben werden. Sie können beispielsweise über zwei Versionen einer Anwendung verfügen, wobei sich die eine Version in der nativen und die andere in der virtuellen Umgebung befindet. Wenn diese Anwendungsversionen Informationen in demselben Speicherobjekt verwenden, können die beiden Anwendungen in Konflikt zueinander geraten und fehlschlagen. Möglicherweise möchten Sie gemeinsam genutzte Arbeitsspeicherobjekte isolieren, um sicherzustellen, dass virtuelle Anwendungen und Systemobjekte sich gegenseitig nicht erkennen können.

Der Parameter `IsolatedMemoryObjects` wird in der Datei `Package.ini` nicht angezeigt, doch Sie können ihn hinzufügen. ThinApp legt einen Anfangswert fest, der die Speicherobjekte isoliert, die von einer nativen Version von Internet Explorer in der virtuellen Umgebung verwendet werden. Mit diesem Wert wird ein Konflikt zwischen den Dienstprogrammen `explorer.exe` und `iexplore.exe` behoben, der beim Zuordnen von Sandbox-Dateien durch die Dienstprogramme auftritt. Mit dem Parameter `IsolatedMemoryObjects` können Sie weitere benannte gemeinsam genutzte Arbeitsspeicherobjekte isolieren, um sicherzustellen, dass die Objekte nur für andere virtuelle Anwendungen sichtbar sind, die dieselbe Sandbox verwenden.

Der Parameter `IsolatedMemoryObjects` akzeptiert eine Liste mit durch Semikolon (;) getrennten Einträgen. Jeder Eintrag kann die Platzhalter Sternchen (*) und Fragezeichen (?) enthalten, um einen Abgleich mit variablen Mustern durchzuführen.

Beispiele

Sie können mit dem Parameter `IsolatedMemoryObjects` festlegen, dass das Speicherobjekt mit dem Namen `My Shared Object` und alle Speicherobjekte, deren Name `outlook` enthält, isoliert werden.

```
[BuildOptions]
IsolatedMemoryObjects=*outlook*;My Shared Object
```

IsolatedSynchronizationObjects

Der Parameter `IsolatedSynchronizationObjects` listet die Synchronisierungsobjekte auf, die von anderen Anwendungen isoliert werden sollen.

Synchronisierungsobjekte dienen zur Koordinierung der Aktionen zwischen Anwendungen. Die folgenden Windows-Synchronisierungsobjekte können in den Protokollen für Anwendungsfehler angezeigt werden:

- `OpenMutex`
- `CreateMutex`
- `OpenSemaphore`
- `CreateSemaphore`

- `OpenEvent`
- `CreateEvent`

Wenn diese Objekte in Protokolldateien angezeigt werden, sollten Sie die Objekte in der virtuellen Umgebung isolieren, um Kollisionen mit den von nativen Anwendungen erstellten Synchronisierungsobjekten zu vermeiden. Sie können Synchronisierungsobjekte von Anwendungen isolieren, die nicht in demselben virtuellen Namensraum ausgeführt werden. Wenn zwei Anwendungen denselben Sandbox-Pfad gemeinsam nutzen, haben die Anwendungen denselben Namensraum für isolierte Synchronisierungsobjekte. Wenn zwei Anwendungen denselben Sandbox-Namen, aber verschiedene Sandbox-Pfade haben, haben die Anwendungen unterschiedliche Namensräume.

Der Parameter `IsolatedSynchronizationObjects` wird in der Datei `Package.ini` nicht angezeigt, doch Sie können ihn hinzufügen. ThinApp legt einen Anfangswert fest, der Synchronisierungsobjekte für andere Anwendungen verfügbar macht. Virtuelle Anwendungen mit unterschiedlichen Sandboxes können die Synchronisierungsobjekte erkennen.

Sie können im Parameter `IsolatedSynchronizationObjects` eine Liste mit durch Semikolons (;) getrennten Einträgen angeben. In den Einträgen können Sternchen (*) und Fragezeichen (?) als Platzhalter verwendet werden, damit sie auf variable Muster zutreffen.

Beispiele

Sie können mit dem Parameter `IsolatedSynchronizationObjects` festlegen, dass das Synchronisierungsobjekt mit dem Namen `My Shared Object` und das Synchronisierungsobjekt, dessen Name `outlook` enthält, isoliert werden.

```
[BuildOptions]
IsolatedSynchronizationObjects=*outlook*;My Shared Object (Mein gemeinsam genutztes Objekt)
```

NotificationDLLs

Der Parameter `NotificationDLLs` ruft DLL-Dateien von Drittanbietern auf, die Benachrichtigungen über Ereignisse, z. B. das Starten oder Beenden einer Anwendung, liefern. Die DLL-Dateien können sich entweder im physischen Dateisystem oder im virtuellen Paket befinden. Wenn ThinApp eine DLL-Datei nicht laden kann, generiert das Paket eine Fehlermeldung.

Dieser Parameter wird in der Datei `Package.ini` nicht angezeigt, doch ThinApp SDK-Benutzer können ihn in der Datei hinzufügen.

Beispiele

Sie können den Parameter `NotificationDLLs` so ändern, dass die Dateien `First.dll` und `Second.dll` aufgerufen werden.

```
[BuildOptions]
NotificationDLLs=First.dll;Second.dll
```

NotificationDLLSignature

Der Parameter `NotificationDLLSignature` überprüft gemeinsam mit dem Parameter `NotificationDLLs`, ob eine bestimmte DLL-Datei über eine Signatur verfügt. Wenn die DLL-Datei keine Signatur hat, wird sie von ThinApp nicht geladen.

Beispiel

Sie können als Wert des Parameters `NotificationDLLSignature` ein Sternchen (*) angeben, um sicherzustellen, dass die DLL-Datei durch den Authentifizierungscode signiert wird.

```
[BuildOptions]
NotificationDLLSignature=*
```

Sie können stattdessen auch eine Entität angeben, um sicherzustellen, dass die DLL-Datei durch diese Entität signiert wird.

```
[BuildOptions]
NotificationDLLSignature=VMware, Inc.
```

ObjectTypes

Der Parameter `ObjectTypes` gibt eine Liste mit virtuellen COM-Objekttypen an, die für andere Anwendungen in der physischen Umgebung sichtbar sind. Sie können Skripte wie z. B. VBScripts verwenden, um Objekte aufzurufen, die gekapselte Anwendungen starten.

Ein Objekttyp kann jeweils nur für eine native oder eine virtuelle Anwendung registriert werden. Wenn Sie Office 2003 auf dem nativen Rechner installieren und ein virtuelles Office 2007-Paket verwenden möchten, müssen Sie festlegen, ob die virtuelle oder die native Anwendung die Objekttypen verarbeiten soll.

Wenn Sie möchten, dass das virtuelle Office 2007 die Objekttypen verarbeitet, können Sie die Einstellung `ObjectTypes` in der `Package.ini`-Datei belassen, das Paket erstellen und es mithilfe des Dienstprogramms `thinreg.exe` registrieren. Wenn Sie möchten, dass das native Office 2003 die Objekttypen verarbeitet, müssen Sie die Einstellung `ObjectTypes` aus der `Package.ini`-Datei entfernen, bevor Sie das Paket erstellen und registrieren. Sie können keine zufälligen Einträge zum `ObjectTypes`-Parameter hinzufügen. Sie können nur Einträge entfernen, die durch den Kapselungsprozess generiert wurden.

Beispiele

Wenn ein Skript oder eine native Anwendung ein COM-Objekt `Excel.Application` oder andere COM-Objekte erstellt, die im Parameter `ObjectTypes` aufgelistet sind, startet ThinApp das virtuelle Paket.

```
[Microsoft Office Excel 2007.exe]
ObjectTypes=Excel.Application;Excel.Application.12;Excel.Chart;
               Excel.Macrosheet;Excel.Sheet; Excel.Workspace
```

SandboxCOMObjects

Der Parameter `SandboxCOMObjects` gibt an, ob Anwendungen in der physischen Umgebung auf COM-Objekte zugreifen können, die während der Laufzeit von der virtuellen Anwendung registriert werden.

ThinApp legt einen Anfangswert fest, der verhindert, dass native Anwendungen in der physischen Umgebung auf COM-Objekte zugreifen, die von der virtuellen Anwendung registriert werden. ThinApp legt in der Sandbox die COM-Objekte ab, die von der virtuellen Anwendung registriert werden.

Beispiele

Sie können den Parameter `SandboxCOMObjects` entsprechend ändern und COM-Objekte sichtbar machen, die von der virtuellen Anwendung außerhalb der Sandbox registriert werden. Wenn Sie beispielsweise eine native Version von Microsoft Office 2003 und eine virtuelle Version von Microsoft Office 2007 installieren und in der nativen Umgebung ein benutzerdefiniertes Programm zur Zusammenführung von E-Mails ausführen, das Microsoft Word startet und es anweist, das Dokument zu öffnen, zu ändern und zu speichern, können Sie Microsoft Word 2007-Dokumente erstellen, während die virtuelle Microsoft Word-Version ausgeführt wird. Die native Anwendung kann auf COM-Objekte aus der virtuellen Anwendung zugreifen.

```
SandboxCOMObjects=0
```

VirtualizeExternalOutOfProcessCOM

Der Parameter `VirtualizeExternalOutOfProcessCOM` steuert, ob COM-Objekte außerhalb der Prozesse in der virtuellen Umgebung ausgeführt werden können. Außerhalb der virtuellen Umgebung erstellte COM-Objekte werden immer in der physischen Umgebung ausgeführt.

Dieser Parameter bezieht sich auf COM-Objekte außerhalb von Prozessen, die nicht zu einem ThinApp-Paket gehören und die nicht in der virtuellen Registrierung eingetragen sind. Der von ThinApp festgelegte Anfangswert für den Parameter `VirtualizeExternalOutOfProcessCOM` gibt an, dass externe COM-Objekte außerhalb von Prozessen in der virtuellen Umgebung ausgeführt werden sollen, um sicherzustellen, dass die COM-Objekte den Hostcomputer nicht modifizieren können. Falls ein Kompatibilitätsproblem mit einem externen COM-Objekt besteht, das in der virtuellen Umgebung ausgeführt wird, können Sie COM-Objekte auf dem Hostsystem erstellen und ausführen. Wenn Sie nur spezifische COM-Objekte außerhalb der virtuellen Umgebung ausführen möchten, können Sie mithilfe des Parameters `ExternalCOMObjects` die CLSID der einzelnen COM-Objekte auflisten.

Beispiele

Sie können den Parameter `VirtualizeExternalOutOfProcessCOM` so ändern, dass alle externen COM-Objekte außerhalb der Prozesse in der physischen Umgebung anstatt in der virtuellen Umgebung ausgeführt werden. So können Sie beispielweise mit einer virtuellen Version von Microsoft Access 2003 E-Mails über eine native Sitzung in IBM Lotus Notes senden.

```
[BuildOptions]
VirtualizeExternalOutOfProcessCOM=0
```

Konfigurieren von Dateispeicher

ThinApp-Parameter können verwendet werden, um Dateispeicher zu konfigurieren und virtuelle Laufwerke einzurichten.

Weitere Informationen zur Speicherung in Zusammenhang mit der Sandbox-Konfiguration finden Sie in „Konfigurieren von Sandbox-Speicher und Bestandsnamen“ auf Seite 96.

CachePath

Der Parameter `CachePath` legt für das Bereitstellungssystem den Pfad eines Cache-Verzeichnisses für Schriftartdateien und ausführbare Stub-Dateien fest.

Da Schriftartdateien und ausführbare Stub-Dateien häufig verwendet werden, muss ThinApp die Dateien im Cache schnell extrahieren und auf der physischen Festplatte platzieren. Wenn Sie den Cache löschen, kann ThinApp ihn wiederherstellen.

Mit der Umgebungsvariablen `THINSTALL_CACHE_DIR` können Sie den Parameter `CachePath` während der Laufzeit außer Kraft setzen. Wenn Sie weder die Umgebungsvariable `THINSTALL_CACHE_DIR` noch den Parameter `CachePath` angeben, legt ThinApp für den Parameter `CachePath` basierend auf dem Parameter `SandboxPath` und gemäß den folgenden Regeln einen Anfangswert fest:

- Wenn der Parameter `SandboxPath` in der Datei `Package.ini` angegeben und auf einen relativen Pfad festgelegt wurde, verwendet der Parameter `CachePath` den Sandbox-Pfad und speichert den Cache auf derselben Verzeichnisebene wie die Sandbox.
- Wenn der Parameter `SandboxPath` in der Datei `Package.ini` angegeben und auf einen absoluten Pfad festgelegt ist oder wenn der Parameter `SandboxPath` in der Datei `Package.ini` nicht angegeben ist, verwendet der Parameter `CachePath` den Speicherort `%Local AppData%\Thinstall\Cache`. In diesem Fall wird das Cache-Verzeichnis auf dem lokalen Rechner platziert, unabhängig davon, wohin der Benutzer die Sandbox verschiebt. ThinApp erstellt innerhalb des Cache das Verzeichnis `Stubs`.

Beispiele

Sie können den Parameter `CachePath` auf einen absoluten Pfad festlegen.

```
CachePath=C:\VirtCache
```

Sie können auch einen relativen Pfad festlegen, der von ThinApp in Relation zu dem Verzeichnis interpretiert wird, in dem die ausführbaren Anwendungsdateien gespeichert sind. Wenn das Paket in `C:\VirtApps` gespeichert ist und der Parameter `CachePath` den Wert `Cache` aufweist, lautet das Cache-Verzeichnis `C:\VirtApps\Cache`.

```
CachePath=Cache
```

Wenn Sie ein USB-Gerät verwenden und die Sandbox auf das USB-Gerät verschieben, sollten Sie den Cache möglicherweise ebenfalls auf das USB-Gerät verschieben, um Konflikte mit dem lokalen Rechner zu vermeiden. Im folgenden Beispiel befinden sich Cache und Sandbox auf derselben Verzeichnisebene.

```
CachePath=<sandbox_path>
```

UpgradePath

Der Parameter `UpgradePath` gibt den Speicherort von Informationen und Dateien für Application Sync und Integer-Updates an.

Der von ThinApp festgelegte Anfangswert legt fest, dass das Dienstprogramm Application Sync seine Protokoll- und Cache-Dateien an demselben Speicherort wie die ausführbare Anwendungsdatei auf dem lokalen Computer speichert. Bei Integer-Updates wird auf ähnliche Weise mit Dateien umgegangen.

Wenn das Dienstprogramm Application Sync ein Update von einem Server herunterlädt, speichert es das Update unter einem temporären Namen an dem Speicherort, der unter `UpgradePath` angegeben ist. Beim nächsten Starten der Anwendung benennt ThinApp die temporäre Datei mit einer Dateierweiterung .1 oder .2 um, je nachdem, ob .1 bereits vorhanden ist. ThinApp versucht den Namen mit der Erweiterung .1 in den Originalnamen der Datei zu ändern, der möglicherweise in einem anderen Verzeichnis gespeichert ist. Wenn ThinApp diese Änderung nicht durchführt, behält die Datei die Erweiterung .1 an dem Speicherort von `UpgradePath` bei. Bei der Ausführung der Originalanwendung wird auf diese Datei zugegriffen.

Informationen über das Dienstprogramm Application Sync erhalten Sie unter „Application Sync-Updates“ auf Seite 43.

Beispiele

Wenn der Standardspeicherort nur über beschränkten Speicherplatz verfügt oder wenn Sie Upgrades von der ausführbaren Anwendungsdatei isolieren möchten, ändern Sie den Parameter `UpgradePath` entsprechend, um einen alternativen Speicherort zum Erkennen von Update-Dateien anzugeben. Der Parameter kann Umgebungsvariablen im Pfad enthalten, unterstützt jedoch keine Ordnermakros.

```
[BuildOptions]
UpgradePath=C:\Programme\<my_app_upgrades>
```

VirtualDrives

Der Parameter `VirtualDrives` gibt zusätzliche Laufwerkbuchstaben an, die während der Laufzeit für die Anwendung verfügbar sind.

ThinApp stellt die virtuelle Umgebung so dar, dass sie der physischen Kapselungsumgebung möglichst ähnlich ist, und bildet die physischen Laufwerke ab, die auf dem Kapselungssystem verfügbar sind. ThinApp verwendet zum Darstellen virtueller Laufwerke den Parameter `VirtualDrives` und einen Projektordner, z. B. `%drive_<Laufwerkbuchstabe>%`, der die virtuellen Dateien im Laufwerk enthält. Dieser Projektordner kann sich im schreibgeschützten Dateisystem des Pakets und in der Sandbox befinden, wenn Schreibvorgänge im physischen Laufwerk nicht ausgeführt werden können.

Der Parameter `VirtualDrives` stellt die Laufwerke während der Laufzeit für die Anwendung dar. Außerdem zeigt der Parameter `VirtualDrives` Metadaten des Laufwerks an, z. B. die Seriennummer und den Typ des Laufwerks. ThinApp erkennt beispielsweise das physische Laufwerk `C:` auf dem Kapselungssystem und gibt im Parameter als Typ des Laufwerks den Wert „FIXED“ sowie die Seriennummer des Laufwerks ein.

Der Parameter `VirtualDrives` enthält die folgenden Informationen:

- Drive – Der Laufwerkbuchstabe, ein einzelnes alphabetisches Zeichen von A bis Z.
- Serial – Die Seriennummer als achtstellige Hexadezimalzahl.

- Type – Der Laufwerkstyp, bezeichnet durch einen der Werte „FIXED“, „REMOVABLE“, „CD-ROM“ oder „RAMDISK“.
  - FIXED – Weist auf fest eingebaute Medien hin.

    Beispiel: eine Festplatte oder ein eingebautes Flash-Laufwerk.
  - REMOVABLE – Weist auf Wechselmedien hin.

    Beispiel: ein Festplattenlaufwerk, ein Thumb-Laufwerk oder ein Flash-Kartenlesegerät.
  - CD-ROM – Weist auf ein CD-ROM-Laufwerk hin.
  - `RAMDISK` – Weist auf eine RAM-Diskette hin.

Virtuelle Laufwerke sind nützlich, wenn Anwendungen auf hart codierte Pfade zu Laufwerkbuchstaben zurückgreifen, die auf den Bereitstellungssystemen möglicherweise nicht verfügbar sind. Zum Beispiel ist es bei älteren Anwendungen wahrscheinlich, dass das Laufwerk `D:` ein CD-ROM-Laufwerk ist und dass Datendateien unter `D:\media` verfügbar sind.

Die physischen Eigenschaften des Laufwerks auf dem physischen Bereitstellungssystem werden durch die virtuellen Laufwerkseinstellungen überschrieben. Wenn der Parameter `VirtualDrives` einer physischen Festplatte den Laufwerkstyp „CD-ROM“ zuweist, erkennt die Anwendung auf dem Bereitstellungssystem dieses Laufwerk als CD-ROM-Laufwerk.

Isolationsmodi für virtuelle Laufwerke

Virtuelle Laufwerke sind nur für diejenigen Anwendungen sichtbar, die in der virtuellen Umgebung ausgeführt werden. Virtuelle Laufwerke wirken sich nicht auf die physische Windows-Umgebung aus. Virtuelle Laufwerke übernehmen Isolationsmodi vom Standardisolationsmodus des Projekts, sofern Sie den Modus nicht mithilfe der Datei `##Attributes.ini` im Laufwerksordner innerhalb des Projektverzeichnisses außer Kraft setzen.

Wenn Sie vor dem Erstellen der Anwendung Dateien in den Ordner `%drive_D%` kopieren, können Sie den Isolationsmodus „Full“ für dieses Laufwerk verwenden. Die Anwendung liest die Daten immer vom virtuellen Laufwerk und versucht nicht, von einem entsprechenden physischen CD-ROM-Laufwerk auf dem Bereitstellungssystem zu lesen.

Wenn Sie vor dem Erstellen der Anwendung keine Dateien in den Ordner `%drive_D%` kopieren, können Sie die Isolationsmodi „Merged“ oder „WriteCopy“ für virtuelle Laufwerksordner verwenden, je nachdem, ob Sie Lese- oder Schreibvorgänge auf dem physischen Laufwerk des Bereitstellungssystems ausführen möchten.

Wenn Sie dem virtuellen Laufwerk den Isolationsmodus „Merged“ zuweisen, dieses Laufwerk auf dem physischen Bereitstellungssystem aber nicht vorhanden ist, schlagen Schreibvorgänge auf diesem Laufwerk fehl. ThinApp leitet Änderungen nicht an die Sandbox um, weil es durch den Isolationsmodus „Merged“ angewiesen wird, den Schreibvorgang auf dem physischen Laufwerk auszuführen. Wenn die Anwendung den Schreibvorgang auf dem physischen Laufwerk nicht gemäß der Anweisung ausführen kann, schlägt der Schreibvorgang fehl.

Die Einstellungen des Isolationsmodus werden durch den Parameter `VirtualDrives` also nicht überschrieben. Deshalb kann es vorkommen, dass eine virtuelle Anwendung Dateien auf einem physischen Laufwerk wegen der Isolationsmoduseinstellungen nicht erkennen kann.

Ändern der Isolationsmodi für virtuelle Laufwerke

Eine Änderung der Isolationsmodi für virtuelle Laufwerke ist nötig, wenn Sie den Standardisolationsmodus des Projekts außer Kraft setzen möchten.

So ändern Sie die Isolationsmodi für ein virtuelles Laufwerk

1 Fügen Sie dem ThinApp-Projekt das Verzeichnis `%Drive_<Laufwerkbuchstabe>%` hinzu.

2 Erstellen Sie die Datei `##Attributes.ini` und fügen Sie ihr einen Eintrag mit dem Isolationsmodus für diesen Laufwerkbuchstaben hinzu.

```
[Isolation]
DirectoryIsolationMode=<Isolationsmodus>
```

3 Platzieren Sie die Datei ##Attributes.ini im Verzeichnis %Drive_<Laufwerkbuchstabe>%.

Beispiele

Der Parameter VirtualDrives ist eine einzelne Zeichenfolge, die Informationen zu mehreren Laufwerkbuchstaben sowie optionale Parameter für die betreffenden Laufwerkbuchstaben enthalten kann. Der Parameter trennt die Informationen, die verschiedenen Laufwerkbuchstaben zugewiesen sind, durch Semikolons; Parameter für einzelne Laufwerkbuchstaben werden durch Kommata getrennt. ThinApp weist dem Laufwerk eine Seriennummer und den Typ FIXED zu.

```
[BuildOptions]
VirtualDrives= Drive=A, Serial=12345678, Type=REMOVABLE; Drive=B, Serial=9ABCDEF0, Type=FIXED
```

Sie können als virtuelle Laufwerkbuchstaben die Buchstaben X, D und Z angeben.

```
[BuildOptions]
VirtualDrives=Drive=X, Serial=ff897828, Type=REMOVABLE; Drive=D, Type=CDROM; Drive=Z
```

Laufwerk X ist ein Wechseldatenträger mit der Seriennummer ff797828.

Laufwerk D ist ein CD-ROM-Laufwerk mit einer zugewiesenen Seriennummer.

Laufwerk Z ist ein FIXED (FESTES) Laufwerk mit einer zugewiesenen Seriennummer.

Konfigurieren von Prozessen und Diensten

Sie können mithilfe der ThinApp-Parameter die Prozesse und Dienste, mit denen einem nativen Prozess Schreibzugriff erteilt werden kann, sowie das Starten und Beenden von virtuellen Diensten konfigurieren.

AllowExternalKernelModeServices

Der Parameter AllowExternalKernelModeServices steuert, ob Anwendungen native Kerneltreiberdienste erstellen und ausführen können. Dazu muss die ausführbare Datei des Dienstes im physischen Dateisystem vorhanden sein.

ThinApp zeigt den Standardparameter nicht in der Datei Package.ini an, weist jedoch einen Anfangswert zu, der das Starten eines nativen Windows-Kerneltreiberdienstes durch die Anwendung verhindert.

Beispiele

Sie können der Datei Package.ini den Parameter AllowExternalKernelModeServices hinzufügen und dessen Standardwert von 0 in 1 ändern, um das Öffnen eines nativen Windows-Kerneltreiberdienstes durch die Anwendung zuzulassen.

```
[BuildOptions]
AllowExternalKernelModeServices=1
```

AllowExternalProcessModifications

Der Parameter AllowExternalProcessModifications legt fest, ob die gekapselten Anwendungen in einen nativen Prozess schreiben können. Für manche virtualisierten Anwendungen ist eine Methode zur Interaktion mit nativen Anwendungen erforderlich.

ThinApp blockiert alle Versuche der gekapselten Anwendung, sich selbst in eine native Anwendung einzuspeisen. Die gekapselte Anwendung kann sich immer noch in virtuelle Anwendungen einfügen, die in derselben Sandbox ausgeführt werden. ThinApp zeigt den Standardparameter nicht in der Package.ini-Datei an.

Wenn ThinApp die Selbsteinspeisung einer gekapselten Anwendung in eine native Anwendung blockiert, generiert Protokoll-Monitor Ablaufverfolgungsprotokolle, die auf den Parameter AllowExternalProcessModifications verweisen.

Beispiele

Sie können der Datei `Package.ini` den Parameter `AllowExternalProcessModifications` hinzufügen, um Schreibvorgänge von virtuellen Prozessen in nativen Prozessen zu unterstützen. Zum Beispiel kann es vorkommen, dass eine Spracherkennungsanwendung sich selbst in native Anwendungen einfügen muss, um Text in Sprache umzuwandeln.

```
[BuildOptions]
AllowExternalProcessModifications=1
```

AllowUnsupportedExternalChildProcesses

Der Parameter `AllowUnsupportedExternalChildProcesses` gibt an, ob untergeordnete 64-Bit-Prozesse in der physischen Umgebung ausgeführt werden dürfen. ThinApp führt 64-Bit-Anwendungen in der physischen Umgebung aus, weil ThinApp 64-Bit-Prozesse nicht unterstützt und 64-Bit-Anwendungen auch nicht virtualisieren kann.

Der von ThinApp für den Parameter `AllowUnsupportedExternalChildProcesses` festgelegte Anfangswert gibt an, dass 64-Bit-Anwendungen in der physischen Umgebung ausgeführt werden. Sie können Aufgaben von untergeordneten 64-Bit-Prozessen ausführen, die auf 64-Bit-Systemen ausgeführt werden. Die Ausführung der Druckwarteschlange ist ein Beispiel für eine Aufgabe eines untergeordneten 64-Bit-Prozesses.

Beispiele

Wenn Sie das physische Dateisystem vor eventuellen Änderungen schützen möchten, können Sie den Parameter `AllowUnsupportedExternalChildProcesses` so ändern, dass das Generieren von untergeordneten 64-Bit-Prozessen durch ThinApp außerhalb der virtuellen Umgebung blockiert wird. ThinApp kann keinerlei 64-Bit-Prozesse ausführen, weil ThinApp diese Prozesse in der virtuellen Umgebung nicht unterstützt.

```
[BuildOptions]
AllowUnsupportedExternalChildProcesses=0
```

AutoShutdownServices

Der Parameter `AutoShutdownServices` steuert, ob virtuelle Dienste beim Beenden des letzten dienstfremden Prozesses heruntergefahren werden.

ThinApp legt einen Anfangswert fest, mit dem virtuelle Dienste gestoppt werden, wenn der letzte dienstfremde Prozess beendet wird. Der Parameter wirkt sich nicht auf Dienste außerhalb des virtuellen Kontexts aus.

Beispiele

Sie können den Parameter `AutoShutdownServices` entsprechend ändern, wenn Sie Apache Web Server ausführen und möchten, dass der virtuelle Dienst auch nach Beendigung der Anwendung, die den Dienst gestartet hat, fortgesetzt wird.

```
[BuildOptions]
AutoShutdownServices=0
```

AutoStartServices

Der Parameter `AutoStartServices` steuert, ob virtuelle Dienste gestartet werden, wenn die erste virtuelle Anwendung gestartet wird.

ThinApp legt einen Anfangswert fest, mit dem virtuelle Dienste gestartet werden, für die bei der Installation der Starttyp „Automatisch“ festgelegt wurde. Die virtuellen Dienste werden gestartet, wenn der Benutzer den ersten übergeordneten Prozess ausführt.

Beispiele

Wenn Anwendungen einen Dienst installieren, jedoch nicht verwenden, können Sie den Parameter `AutoStartServices` so ändern, dass der Start des virtuellen Dienstes verhindert und auf diese Weise Zeit gespart wird.

```
[BuildOptions]
AutoStartServices=0
```

ChildProcessEnvironmentDefault

Der Parameter `ChildProcessEnvironmentDefault` legt fest, ob ThinApp alle untergeordneten Prozesse in der virtuellen Umgebung ausführt.

ThinApp erstellt alle untergeordneten Prozesse in der virtuellen Umgebung. Bei langsamen Prozessen kann es sinnvoll sein, die untergeordneten Prozesse in die physische Umgebung zu verschieben. Als untergeordneter Prozess kann Microsoft Outlook die Leistung beeinträchtigen, wenn beispielsweise das gesamte Postfach in die virtuelle Umgebung kopiert wird.

Mit dem Parameter `ChildProcessEnvironmentExceptions` können Sie spezifische Ausnahmen erstellen. Siehe „ChildProcessEnvironmentExceptions“ auf Seite 77.

Beispiele

Wenn untergeordnete Prozesse nicht in der virtuellen Umgebung ausgeführt werden sollen, um Auswirkungen auf die Verarbeitungsgeschwindigkeit der virtuellen Umgebung zu vermeiden, können Sie den Parameter `ChildProcessEnvironmentDefault` so ändern, dass untergeordnete Prozesse stattdessen in der physischen Umgebung erstellt werden.

```
[BuildOptions]
ChildProcessEnvironmentDefault=External (Extern)
```

ChildProcessEnvironmentExceptions

Der Parameter `ChildProcessEnvironmentExceptions` gibt die untergeordneten Prozesse an, die Ausnahmen zu der Einstellung des Parameters `ChildProcessEnvironmentDefault` bilden.

Wenn Sie den Parameter `ChildProcessEnvironmentDefault`, der die Standardeinstellung angibt, auf `Virtual` setzen, listet der Parameter `ChildProcessEnvironmentExceptions` die Anwendungen auf, die außerhalb der virtuellen Umgebung ausgeführt werden. Wenn Sie den Parameter `ChildProcessEnvironmentDefault` hingegen auf `External` setzen, listet der Parameter `ChildProcessEnvironmentExceptions` die Anwendungen auf, die innerhalb der virtuellen Umgebung ausgeführt werden.

Beispiele

Sie können Ausnahmen für die Ausführung von untergeordneten Prozessen in der virtuellen Umgebung angeben. Wenn die virtuelle Anwendung einen untergeordneten Prozess `notepad.exe` startet, wird der untergeordnete Prozess außerhalb der virtuellen Umgebung ausgeführt.

```
[BuildOptions]
ChildProcessEnvironmentExceptions=AcroRd.exe;notepad.exe
ChildProcessEnvironmentDefault=Virtual (Virtuell)
```

Konfigurieren von Größen

Mithilfe von ThinApp-Parametern können Datei- und Blockgrößen von Anwendungen komprimiert werden.

BlockSize

Der Parameter `BlockSize` steuert die Größe der Komprimierungsblöcke nur dann, wenn ThinApp Dateien für einen Build komprimiert.

Durch eine größere Blockgröße kann eine höhere Komprimierung erzielt werden. Allerdings kann die Leistung durch eine höhere Blockgröße sinken. Dies hat folgende Gründe:

- Der Build-Prozess verlangsamt sich bei größeren Blockgrößen.
- Die Startdauer und die Lesevorgänge für Anwendungen verlangsamen sich bei größeren Blockgrößen.
- Während der Laufzeit ist mehr Arbeitsspeicher erforderlich, wenn größere Blockgrößen verwendet werden.

Der Parameter `BlockSize` kann in der Datei `Package.ini` und in der Datei `##Attributes.ini` angegeben werden. Sie können verschiedene Blockgrößen für unterschiedliche Verzeichnisse innerhalb desselben Projekts verwenden.

Beispiele

Sie können die Standardgröße von 64 KB im Parameter `BlockSize` erhöhen. Folgende Blockgrößen werden unterstützt: 128 KB, 256 KB, 512 KB und 1 MB. Sie können `k` an die Zahl anhängen, um Kilobyte anzugeben, oder `m` zur Angabe von Megabyte.

```
[Compression]
BlockSize=128k
```

CompressionType

Der Parameter `CompressionType` kann alle Dateien in einem Paket komprimieren, mit Ausnahme von portablen ausführbaren Dateien.

Die Komprimierung von Dateien ist sinnvoll, wenn Sie große Pakete haben und die Einsparung von Speicherplatz höchste Priorität hat. Die Komprimierung hat eine schnelle Dekomprimierungsrate und wirkt sich nur unwesentlich auf die Startdauer der meisten Anwendungen und auf den Arbeitsspeicherverbrauch während der Laufzeit aus. Durch die Komprimierung werden ähnliche Komprimierungsverhältnisse erzielt wie mit dem ZIP-Algorithmus.

In Tabelle 5-1 werden Beispiele für Komprimierungsverhältnisse und Startdauer für ein Microsoft Office 2003-Paket aufgelistet, das von einer lokalen Festplatte ausgeführt wird.

Tabelle 5-1. Beispiele für Komprimierungsverhältnisse und Startdauer

Komprimierungstyp	None (Ohne)	Fast (Schnell)
Größe	448.616 KB	257.373 KB
Komprimierungsverhältnis	100 %	57 %
Startdauer (erste Ausführung)	6 Sekunden	6 Sekunden
Startdauer (zweite Ausführung)	0,1 Sekunden	1 Sekunde
Build-Zeit (erste Erstellung)	3 Minuten	19 Minuten
Build-Zeit (zweite Erstellung)	2 Minuten	1,2 Minuten

Die Komprimierung hat Auswirkungen auf die Leistung und kann die Startdauer auf älteren Computern, oder wenn Sie die Anwendung mehrere Male starten und für die Bereitstellung der Daten für die einzelnen Startvorgänge auf den Windows-Festplatten-Cache angewiesen sind, beeinträchtigen.

Der Parameter `CompressionType` hat keine Auswirkungen auf MSI-Dateien. Informationen zur Komprimierung von MSI-Dateien finden Sie unter „MSICompressionType“ auf Seite 79.

Beispiele

ThinApp legt für die Parameter `OptimizeFor` und `CompressionType` Standardwerte fest, die zusammen für maximale Speicherleistung und Startdauer sorgen. ThinApp speichert alle Daten in unkomprimiertem Format.

```
[Compression]
CompressionType=None

[BuildOptions]
OptimizeFor=Memory
```

Diese Konfiguration können Sie verwenden, wenn die Einsparung von Speicherplatz nur mittlere Priorität hat. ThinApp speichert ausführbare Dateien in unkomprimiertem Format, komprimiert jedoch alle anderen Daten.

```
[Compression]
CompressionType=Fast

[BuildOptions]
OptimizeFor=Memory
```

Diese Konfiguration können Sie verwenden, wenn die Einsparung von Speicherplatz höchste Priorität hat. ThinApp komprimiert alle Dateien.

```
[Compression]
CompressionType=Fast

[BuildOptions]
OptimizeFor=Disk
```

MSICompressionType

Der Parameter `MSICompressionType` legt fest, ob MSI-Dateien für die Paketverteilung komprimiert werden. Durch die Komprimierung wird die Leistung beim Öffnen von MSI-Dateien und bei Verwendung von ThinApp SDK verbessert.

Wenn Sie während des Kapselungsrozesses eine MSI-Datei erstellen, fügt ThinApp der Datei `Package.ini` den Parameter `MSICompressionType` hinzu und legt als Anfangswert `Fast` fest, damit die Datei komprimiert wird. Die Dekomprimierung wird zum Zeitpunkt der Installation ausgeführt.

Wenn Sie den Parameter `CompressionType` auf `Fast` gesetzt haben, brauchen Sie den Parameter `MSICompressionType` nicht auch noch auf `Fast` zu setzen. Durch das Festlegen beider Parameter wird die Komprimierungsrate nicht erhöht.

Beispiele

Wenn Sie mit großen Builds arbeiten und die Leistung keine Priorität hat, können Sie den Parameter `MSICompressionType` so ändern, dass MSI-Dateien nicht komprimiert werden.

```
[Compression]
MSICompressionType=none
```

OptimizeFor

Der Parameter `OptimizeFor` steuert, ob ausführbare Dateien komprimiert werden sollen oder ob die Arbeitsspeichernutzung und die Verwendung von Auslagerungsdateien auf der Festplatte reduziert werden sollen, um die Leistung beim Systemstart zu verbessern. Dieser Parameter dient zusammen mit dem Parameter `CompressionType` zur Anpassung von Paketgröße, Speicherzuweisung und Startdauer der Anwendungen.

VMware empfiehlt, die Standardeinstellung der Parameter `OptimizeFor` und `CompressionType` beizubehalten, um die Leistung beim Systemstart und die Arbeitsspeichernutzung zu verbessern. Sie können die Parameter ändern, um geringere Paketgrößen zu erzielen, wenn die Festplattengröße zu den Hauptproblemen zählt. ThinApp komprimiert ausführbare Dateien nur, wenn Sie den Parameter `OptimizeFor` auf `Disk` und den Parameter `CompressionType` auf `Fast` setzen. Ausführbare Dateien, die in komprimiertem Format innerhalb eines Pakets gespeichert sind, können die Leistung und Arbeitsspeichernutzung beeinträchtigen. Wenn ThinApp ausführbare Dateien aus dem komprimierten Format lädt, kann der Dateispeicher nicht für ähnliche Anwendungssuites oder, in einer Umgebung mit mehreren Benutzern wie Terminal Server, nicht für andere Benutzer freigegeben werden.

Wenn Sie alle Paketdateien mit Ausnahme von portablen ausführbaren Dateien komprimieren möchten, können Sie den Standardwert für den Parameter `OptimizeFor` beibehalten und lediglich den Parameter `CompressionType` auf `Fast` setzen. Wenn Sie hingegen nur MSI-Dateien komprimieren möchten, verwenden Sie den Parameter `MSICompressionType`.

Beispiele

Für maximale Leistung empfiehlt VMware die Standardkonfiguration der Parameter `OptimizeFor` und `CompressionType`. ThinApp speichert alle Daten in unkomprimiertem Format.

```
[Compression]
CompressionType=None

[BuildOptions]
OptimizeFor=Memory
```

VMware empfiehlt folgende Konfiguration, wenn die Einsparung von Speicherplatz mittlere Priorität hat. ThinApp speichert ausführbare Dateien in unkomprimiertem Format, komprimiert jedoch alle anderen Daten.

```
[Compression]
CompressionType=Fast

[BuildOptions]
OptimizeFor=Memory
```

VMware empfiehlt folgende Konfiguration, wenn die Einsparung von Speicherplatz höchste Priorität hat. ThinApp komprimiert alle Dateien.

```
[Compression]
CompressionType=Fast

[BuildOptions]
OptimizeFor=Disk
```

Konfigurieren der Protokollierung

Mithilfe von ThinApp-Parametern können Sie Protokollierungsaktivitäten verhindern oder den Speicherort der Protokolldateien anpassen.

DisableTracing

Der Parameter `DisableTracing` verhindert die Generierung von `.trace`-Dateien, wenn Sie Protokoll-Monitor zur Wahrung von Sicherheit und Ressourcenintegrität ausführen.

Sie können die standardmäßige Generierung der Datei `.trace` blockieren, damit der Anwendungsverlauf nicht für Benutzer einsehbar ist. In einer Testumgebung können Sie die Ablaufverfolgung für bestimmte Anwendungen ausschalten, von denen Sie wissen, dass sie ordnungsgemäß funktionieren. Durch die Erzeugung überflüssiger `.trace`-Dateien wird unnötig viel Festplattenspeicher und CPU-Zeit verbraucht.

Beispiele

Sie können den Parameter `DisableTracing` entsprechend ändern, um die Generierung von `.trace`-Dateien in Protokoll-Monitor zu verhindern.

```
[BuildOptions]
DisableTracing=1
```

LogPath

Der Parameter `LogPath` legt den Speicherort fest, an dem `.trace`-Dateien während der Protokollierungsaktivität gespeichert werden sollen.

Der Standardspeicherort ist dasselbe Verzeichnis, in dem auch die ausführbare Anwendungsdatei gespeichert wird. Sie können den Standardspeicherort ändern, um ein Verzeichnis mit größerem Speicherplatz zu verwenden oder, um die Protokolle von einem USB-Gerät auf den Clientcomputer umzuleiten. Im Gegensatz zu den meisten Pfaden in ThinApp darf der Protokollpfad keine Makros wie `%AppData%` oder `%Temp%` enthalten.

Beispiele

Sie können mit dem Parameter `LogPath` festlegen, dass Protokolldateien im Verzeichnis `C:\ThinappLogs` gespeichert werden sollen.

```
[BuildOptions]
LogPath=C:\ThinappLogs
```

Konfigurieren von Versionen

ThinApp-Parameter bieten Informationen über die Versionen der ausführbaren Anwendungsdateien und ThinApp.

CapturedUsingVersion

Der Parameter `CapturedUsingVersion` zeigt die Version von ThinApp für den Kapselungsprozess an und bestimmt die Dateisystemmakros, die von ThinApp erweitert werden müssen.

Der in der Datei `Package.ini` angegebene Parameter darf nicht geändert oder gelöscht werden. ThinApp verwendet diesen Parameter für Abwärtskompatibilität und technischen Support.

Beispiele

Der `CapturedUsingVersion` kann z. B. Folgendes anzeigen: ThinApp version 4.0.0-2200.

```
[BuildOptions]
CapturedUsingVersion=4.0.0–2200
```

StripVersionInfo

Der Parameter `StripVersionInfo` gibt an, ob alle Versionsinformationen aus der ausführbaren Quelldatei entfernt werden sollen, wenn ThinApp die Anwendung erstellt. Die ausführbare Quelldatei ist die im Parameter `Source` aufgeführte Datei.

Die Versionsinformationen für ausführbare Dateien werden in den Windows-Eigenschaften angezeigt. Die Informationen über Eigenschaften umfassen Informationen zum Urheberrecht, zu Marken und zur Versionsnummer. Mit dem Parameter `StripVersionInfo` kann die Registerkarte **Version** aus den Windows-Eigenschaften entfernt werden.

ThinApp legt für den Parameter `StripVersionInfo` einen Anfangswert fest, mit dem alle Versionsinformationen in der ausführbaren Quelldatei kopiert werden.

Beispiele

In seltenen Fallen kann es sinnvoll sein, den Parameter `StripVersionInfo` so zu ändern, dass eine Anwendung ohne Versionsinformationen generiert wird. Dies kann beispielsweise der Fall sein, wenn Sie Scanvorgänge zur Versionserkennung umgehen möchten, bei denen die Versionen mit einer Datenbank veralteter Software abgeglichen werden.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
StripVersionInfo=1
```

Version.XXXX

Der Parameter `Version.XXXX` setzt in den Windows-Eigenschaften auf der Registerkarte **Version** Versionszeichenfolgen von Anwendungen außer Kraft oder fügt neue Versionszeichenfolgen hinzu.

Dieser Parameter wird nicht durch den Kapselungsprozess generiert. Sie können diesen Parameter in der Datei `Package.ini` hinzufügen.

Beispiele

Sie können mit dem Parameter `Version.XXXX` einen neuen Produktnamen festlegen. Beispielsweise können Sie `ThinApp Office` anstatt `Office` als Produktnamen festlegen. Geben Sie den Wert in folgendem Format an: `Version.<Zeichenfolgenname>=<Zeichenfolgenwert>`.

```
[<Anw>.exe]
Version.ProductName=ThinApp Office
Version.Description=This Product is great! (Dieses Produkt ist fantastisch!)
```

Konfigurieren von Gebietsschemata

ThinApp-Parameter können verwendet werden, um Gebietsschemainformationen zu überprüfen.

AnsiCodePage

Der Parameter `AnsiCodePage` gibt einen numerischen Wert an, der für die Sprache des Betriebssystems steht, unter dem Sie die Anwendung kapseln. ThinApp verwendet diesen Wert, um Multibyte-Zeichenfolgen zu verwalten.

Dieser Parameter führt keine Sprachübersetzungen aus. Der Wert wirkt sich lediglich auf die Anzeige von Textzeichenfolgen und die Verwendung von Zeichenfolgen innerhalb der Anwendung aus.

Beispiele

Wenn die Betriebssysteme des Bereitstellungs- und Kapselungscomputers unterschiedliche Sprachen haben, können Sie den Parameter `AnsiCodePage` überprüfen.

```
[BuildOptions]
AnsiCodePage=1252
```

LocaleIdentifier

Der Parameter `LocaleIdentifier` zeigt eine numerische ID für das Gebietsschema an, das sich auf Layout und Formatierung auswirkt. Der Wert sucht in der Anwendung nach den entsprechenden Sprachressourcen.

Bei der Ausführung von Paketen richtet sich ThinApp nach den Regions- und Spracheinstellungen des Kapselungssystems und nicht nach den Einstellungen des Systems, auf dem die Pakete ausgeführt werden. Wenn Sie eine Anwendung, die ein Gebietsschemaformat, z. B. ein Datumsformat, erfordert, auf einem System kapseln, das nicht über das erforderliche Format verfügt, können Sie diesen Parameter auskommentieren, um sicherzustellen, dass die Anwendung auf einem System ausgeführt werden kann, das über das unterstützte Format verfügt.

Beispiele

Wenn die Regionssprache des Betriebssystems auf „Englisch (USA)“ eingestellt ist, setzt der Kapselungsprozess den Parameter `LocaleIdentifier` auf den Wert 1033.

```
[BuildOptions]
LocaleIdentifier=1033
```

LocaleName

Der Parameter `LocaleName` zeigt den Namen für das Gebietsschema an, wenn Sie eine Anwendung in Microsoft Vista kapseln.

Beispiele

Der Parameter `LocaleName` kann den Wert für das japanische Gebietsschema anzeigen.

```
[BuildOptions]
LocaleName=ja-JP
```

Konfigurieren von einzelnen Anwendungen

Mithilfe von ThinApp-Parametern können Sie bestimmte Anwendungen konfigurieren.

Die für bestimmte Einstiegspunkte spezifischen Parameter sind in der Datei `Package.ini` unter den Überschriften für die jeweiligen Anwendungen, die das Format `[<Anwendung>.exe]` haben, angegeben. Zum Beispiel wirken sich Einträge unter `[Adobe Reader 8.exe]` für eine Adobe Reader-Anwendung möglicherweise auf die Bereiche Befehlszeilenargumente und Anwendungsverknüpfungen aus.

CommandLine

Der Parameter `CommandLine` gibt die Befehlszeilenargumente an, mit denen eine ausführbare Verknüpfungsdatei gestartet wird. Während der Parameter `Source` den Pfad zur ausführbaren Verknüpfungsdatei angibt, gibt der Parameter `CommandLine` die Datei mit den erforderlichen Optionen oder Parametern zum Starten der ausführbaren Datei an.

Wenn die **Startmenü**-Verknüpfung der Anwendung Befehlszeilenoptionen enthält, legt der Kapselungsprozess den Anfangswert des Parameters `CommandLine` basierend auf diesen Optionen fest. In seltenen Fällen der Problembehandlung müssen Sie diesen Parameter gegebenenfalls auf Anweisung des technischen Supports ändern.

Die Optionen und Parameter folgen dem Basisanwendungsnamen. Verwenden Sie, abhängig von der Anwendung, / oder – vor der Option oder dem Parameter. Verwenden Sie Ordnermakros für die Pfadbenennungskonventionen.

Beispiele

Sie können für den Parameter `CommandLine` einen Eintrag basierend auf der**Startmenü**-Verknüpfung `"C:\Programme\Mozilla Firefox\firefox.exe" –safe–mode` erstellen.

```
CommandLine="C:\Program Files\Mozilla Firefox\firefox.exe" –safe–mode
```

Befehlszeilenargumente werden in folgendem Format angegeben: `/<Option> <Parameter>`.

```
[<Anw>.exe]
Source=%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe
Shortcut=<Primärer_Datenbehälter>.exe
CommandLine="%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe" /<Option> <Parameter>
```

Disabled

Der Parameter `Disabled` legt fest, ob das Build-Ziel einer Anwendung nur ein Platzhalter ist, und verhindert in diesem Fall, dass ThinApp die ausführbare Datei im Verzeichnis `/bin` generiert.

ThinApp aktiviert Einstiegspunkte, wenn Verknüpfungen des Installationsprogramms für eine Anwendung auf dem Desktop und im **Startmenü** vorhanden sind. Wenn Sie keinen der im Setup Capture-Assistenten angezeigten Einstiegspunkte auswählen, legt ThinApp für den Parameter `Disabled` einen Anfangswert fest, der verhindert, dass die ausführbare Datei der Anwendung während des Build-Prozesses generiert wird.

Beispiele

Wenn Sie die Einstiegspunkte `cmd.exe`, `regedit.exe` oder `iexplore.exe` für die Fehlerbehebung nicht während des Kapselungsprozesses auswählen und Sie zu einem späteren Zeitpunkt einen Debug-Vorgang für die Umgebung ausführen müssen, können Sie den Parameter `Disabled` entsprechend ändern, damit diese Einstiegspunkte generiert werden.

```
[app.exe]
Source=%ProgramFilesDir%\<Meine_Anw>\<Anw>.exe
Disabled=0
```

ReadOnlyData

Der Parameter `ReadOnlyData` gibt den Namen der schreibgeschützten virtuellen Registrierungsdatei an, die bei der Erstellung der Anwendung erstellt wird, und weist den primären Datencontainer für eine Anwendung zu.

Dieser Parameter darf nicht geändert werden. Für den Fall, dass Sie den Speicherort des primären Datencontainers ermitteln müssen, wird der Parameter in der Datei `Package.ini` angezeigt.

Wenn der primäre Datencontainer kleiner als 200 MB ist, wird der Container in einer ausführbaren Datei mit Einstiegspunkt gespeichert. Ist die primäre Datencontainerdatei größer als 200 MB, speichert ThinApp den Container als `.dat`-Datei, die nicht als Einstiegspunkt für die Anwendung dienen kann.

Beispiele

ThinApp legt für den Parameter `ReadOnlyData` den erforderlichen Wert `Package.ro.tvr` als Namen der virtuellen Registrierungsdatei fest.

```
ReadOnlyData=bin\Package.ro.tvr
```

ReserveExtraAddressSpace

Der Parameter `ReserveExtraAddressSpace` gibt die Größe des zusätzlichen Adressbereichs an, der für die gekapselte ausführbare Datei reserviert werden soll.

ThinApp bestimmt anhand der ausführbaren Datei, die im Parameter `Source` angegeben ist, die erforderliche Größe des Arbeitsspeicheradressbereichs für eine Anwendung. Wenn Sie ein Paket erstellen, zu dem eine ausführbare Datei gehört, die jedoch nicht im Paket selbst enthalten ist, wie es z. B. beim Einstiegspunkt `cmd.exe` der Fall ist, oder wenn Sie für eine Anwendung eine automatische Update-Funktion aktivieren, die die neue Version der Anwendung an die Sandbox umleitet, stellt ThinApp entsprechend dem höheren Bedarf zwar mehr Arbeitsspeicher zur Verfügung, reserviert jedoch keine feste Größe an zusätzlichem Adressbereich. In seltenen Fällen können Sie vom technischen Support angewiesen werden, mithilfe des Parameters `ReserveExtraAddressSpace` Arbeitsspeicherplatz hinzuzufügen.

Beispiele

Sie können das Windows-Ladeprogramm anweisen, einen zusätzlichen Adressbereich zu reservieren. Hängen Sie `k` an die Zahl an, um Kilobyte anzugeben, oder `m` zur Angabe von Megabyte.

```
[<Anw>.exe]
Source=%ProgramFilesDir%\<Eigene_Anw>\<Anw>.exe
ReserveExtraAddressSpace=512K
```

Shortcut

Der Parameter `Shortcut` verweist eine ausführbare Verknüpfungsdatei auf einen primären Datencontainer, der das virtuelle Dateisystem und die virtuelle Registrierung enthält. Ein primärer Datencontainer kann in der `Package.ini`-Datei von anderen Einstiegspunkten unterschieden werden, da der primäre Datencontainer den `ReadOnlyData`-Eintrag enthält und die anderen Einstiegspunkte den `Shortcut (Verknüpfung)`-Eintrag enthalten.

Die verknüpfte ausführbare Datei muss im selben Verzeichnis gespeichert sein wie die primäre Datencontainerdatei, damit die Anwendung gestartet werden kann. Informationen über den primären Datencontainer erhalten Sie unter „ReadOnlyData“ auf Seite 84.

Der Wert des Parameters `Shortcut` darf nicht geändert werden. ThinApp erkennt den primären Datencontainer während des Kapselungsprozesses.

Beispiele

ThinApp kann die verknüpfte ausführbare Datei `AcroRd32.exe` auf die primäre Datencontainerdatei `Adobe Reader 8.exe` verweisen.

```
[AcroRd32.exe]
Shortcut=Adobe Reader 8.exe
Source=%ProgramFilesDir%\Adobe\Reader 8.0\Reader\AcroRd32.exe
```

ThinApp kann die verknüpfte ausführbare Datei `Microsoft Office Word 2007.exe` auf die primäre Datencontainerdatei `Microsoft Office Enterprise 2007.dat` verweisen.

```
[Microsoft Office Word 2007.exe]
Source=%ProgramFilesDir%\Microsoft Office\Office12\WINWORD.EXE
Shortcut=Microsoft Office Enterprise 2007.dat
```

Shortcuts

Der Parameter `Shortcuts` listet die Speicherorte auf, an denen das Dienstprogramm `thinreg.exe` eine Verknüpfung zu einer virtuellen Anwendung erstellt.

Der Kapselungsprozess bestimmt `Shortcuts (Verknüpfungen)`-Einträge basierend auf den Verknüpfungen, die das Installationsprogramm der Anwendung implementiert. MSI-Dateien verwenden den Parameter `Shortcuts (Verknüpfungen)`, um die Verknüpfungen zu bestimmen, die erstellt werden sollen.

Beispiele

Sie können den Parameter `Shortcuts` so ändern, dass im Microsoft Office-Ordner im **Startmenü** eine Verknüpfung zur Anwendung Microsoft Word 2003 erstellt wird. Verwenden Sie zum Trennen der Einträge ein Semikolon, wenn Sie Speicherorte für Verknüpfungen hinzufügen. Jeder Eintrag kann Ordnermakros enthalten.

```
[Microsoft Office Word 2003.exe]
ReadOnlyData=bin\Package.ro.tvr
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\WINWORD.EXE
Shortcuts=%Programs%\Microsoft Office
```

Source

Der Parameter `Source` gibt die ausführbare Datei an, die ThinApp lädt, wenn Sie eine ausführbare Verknüpfungsdatei verwenden. Der Parameter liefert den Pfad zur ausführbaren Datei im virtuellen oder physischen Dateisystem.

ThinApp gibt die Quelle für die einzelnen ausführbaren Dateien an. Wenn eine Anwendungssuite drei Benutzereinstiegspunkte hat, zum Beispiel `Winword.exe`, `Powerpnt.exe` und `Excel.exe`, werden in der `Package.ini`-Datei drei Anwendungseinträge aufgelistet. Jeder Eintrag besitzt einen eindeutigen Quelleintrag.

Kann ThinApp die ausführbare Quelldatei nicht im virtuellen Dateisystem lokalisieren, durchsucht ThinApp das physische Dateisystem. Wenn Sie beispielsweise den nativen Internet Explorer aus der virtuellen Umgebung verwenden, lädt ThinApp die ausführbare Quelldatei aus dem physischen Dateisystem.

Zwischen dem Parameter `Source` und dem Verzeichnis `/bin` besteht keinerlei Beziehung. Während im Verzeichnis `/bin` die generierte ausführbare Datei gespeichert wird, verweist der Parameter `Source` auf die installierte ausführbare Datei, die im schreibgeschützten virtuellen Dateisystem gespeichert ist.

Der im Parameter `Source` angegebene Pfad darf nicht geändert werden. Der Kapselungsprozess bestimmt den Pfad basierend darauf, wo das Installationsprogramm die ausführbare Datei im physischen Dateisystem des Computers, mit dem die Kapselung ausgeführt wurde, speichert. ThinApp erstellt einen Pfad für das virtuelle Dateisystem basierend auf dem physischen Dateisystempfad.

Beispiele

Der Parameter `Source` kann auf einen Einstiegspunkt in `C:\Programme\<Basisanw>\<Anw>.exe` verweisen.

```
[<Anw>.exe]
Source=%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe
```

WorkingDirectory

Der Parameter `WorkingDirectory` gibt den ersten Speicherort an, an dem eine Anwendung nach Dateien sucht und Dateien platziert.

ThinApp schließt diesen Parameter nicht standardmäßig in die `Package.ini`-Datei ein, da ThinApp davon ausgeht, dass das Arbeitsverzeichnis dasjenige Verzeichnis ist, in dem sich die ausführbare Datei befindet. Der typische Speicherort in einer ThinApp-Umgebung ist auf dem Desktop des Arbeitscomputers.

Sie können das Arbeitsverzeichnis für einzelne Anwendungen festlegen. Das Arbeitsverzeichnis kann – abhängig von der Isolationsmoduseinstellung – in einem virtuellen Dateisystem, in der Sandbox oder im physischen System vorhanden sein. Sie können Ordnermakros für die Pfadbenennungskonventionen verwenden.

Der Parameter `WorkingDirectory` legt den anfänglichen Wert des Arbeitsverzeichnisses fest, doch das Verzeichnis ist dynamisch, wenn Sie zu anderen Speicherorten navigieren.

Beispiele

Wenn Sie eine Anwendung auf einem USB-Laufwerk haben, können Sie im Parameter `WorkingDirectory` statt des standardmäßigen USB-Speicherorts das Verzeichnis `Eigene Dateien` auf dem Desktop angeben.

```
[<Anw>.exe]
WorkingDirectory=%Personal%
```

Der Speicherort des Verzeichnisses `Eigene Dateien` hängt von der Isolationsmoduseinstellung ab. Wenn Sie eine Zuordnung zwischen dem Arbeitsverzeichnis und dem Verzeichnis `Eigene Dateien` erstellen möchten, geben Sie die Isolationsmoduseinstellung „Merged“ an. Wenn Sie eine Zuordnung zwischen dem Arbeitsverzeichnis und der Sandbox auf dem lokalen Computer erstellen möchten, geben Sie die Isolationsmoduseinstellung „WriteCopy“ oder „Full“ an.

Konfigurieren von abhängigen Anwendungen mit Application Link

Das Dienstprogramm Application Link unterteilt gemeinsam genutzte Komponenten oder abhängige Anwendungen in separate Pakete. Mithilfe der Einträge `OptionalAppLinks` und `RequiredAppLinks` in der Datei `Package.ini` können Sie ThinApp-Pakete dynamisch während der Laufzeit auf Endbenutzercomputern kombinieren. Durch diesen Prozess können Sie Komponententeile separat verpacken, implementieren und aktualisieren und dabei die Vorteile der Virtualisierung von Anwendungen erhalten.

ThinApp kann bis zu 250 Pakete gleichzeitig kombinieren. Die Größe eines jeden Pakets ist beliebig und die Verknüpfungen müssen auf den primären Datencontainer eines Pakets verweisen.

Sandbox-Änderungen von verknüpften Paketen sind für das Basispaket nicht sichtbar. Sie können zum Beispiel Acrobat Reader als eigenständiges virtuelles Paket installieren und als mit der Basisanwendung Firefox verknüpftes Paket. Wenn Sie Acrobat Reader als eigenständige Anwendung starten, indem Sie das virtuelle Paket ausführen und Änderungen in den Voreinstellungen vornehmen, speichert ThinApp die Änderungen in der Sandbox für Acrobat Reader. Wenn Sie Firefox starten, kann Firefox diese Änderungen nicht erkennen, weil Firefox eine eigene Sandbox hat. Beim Öffnen einer `PDF`-Datei mit Firefox werden die Änderungen der Voreinstellungen, die in der eigenständigen Anwendung Acrobat Reader vorhanden sind, nicht angezeigt.

Weitere Informationen über das Dienstprogramm Application Link erhalten Sie unter „Application Link-Updates“ auf Seite 47, „OptionalAppLinks“ auf Seite 88 und „RequiredAppLinks“ auf Seite 87.

Pfadnamenformate für Application Link

Das Dienstprogramm Application Link unterstützt die folgenden Pfadnamenformate:

- Pfadnamen können sich auf die ausführbare Basisdatei beziehen. Beispielsweise verweist `RequiredAppLinks=..\SomeDirectory` auf `c:\MyDir\SomeDirectory`, wenn Sie die ausführbare Basisdatei im Pfad `c:\MyDir\SubDir\ Dependency.exe` bereitstellen.

- Als Pfadnamen können absolute Pfadnamen verwendet werden. Beispiel: `RequiredAppLinks=c:\SomeDirectory`.
- Pfadnamen können eine Netzwerkfreigabe oder einen UNC-Pfad verwenden. Beispiel: `RequiredAppLinks=\\share\somedir\Dependency.exe`.
- Pfadnamen können Umgebungsvariablen enthalten und dynamisch auf alle vorstehenden Pfadnamen erweitert werden. Beispiel: `RequiredAppLinks=%MyEnvironmentVariable%\Package.dat`.

  Bei der Verwendung von Umgebungsvariablen besteht die Gefahr, dass ein Benutzer vor dem Start der Anwendung die Werte ändert und dadurch eine andere Application Link-Abhängigkeit erstellt, als der Administrator eingerichtet hat.
- Pfadnamen können ThinApp-Ordnermakros enthalten. Beispiel: `RequiredAppLinks=%SystemSystem%\Package.dat`.
- Pfadnamen können mehrere Links oder Abhängigkeiten durch ein Semikolon angeben, mit dem einzelne Dateinamen voneinander getrennt werden. Beispiel: `RequiredAppLinks=Dependency1.exe; Dependency2.exe;`.

RequiredAppLinks

Der Parameter `RequiredAppLinks` gibt eine Liste mit erforderlichen Paketen an, die während der Laufzeit in das Basispaket importiert werden sollen. Sie können diesen Parameter in der `Package.ini`-Datei des Basispakets konfigurieren.

Schlägt der Importvorgang für ein abhängiges Paket fehl, so wird eine Fehlermeldung angezeigt und die ausführbare Basisdatei wird beendet. Sie können den Vorgang stattdessen mit dem Parameter `OptionalAppLinks` fortsetzen, selbst wenn Fehler beim Laden auftreten. Wenn Sie ein Paket mithilfe eines Platzhaltermusters angeben und Dateien nicht mit dem Platzhaltermuster übereinstimmen, generiert ThinApp keine Fehlermeldung.

Zum Importieren von Paketen müssen folgende Vorgänge ausgeführt werden:

- Ausführen aus VBScripts aus importierten Paketen
- Starten von Autostart-Diensten aus importierten Paketen
- Registrieren von Schriftarten aus importierten Paketen
- Verschieben der SxS DLL-Dateien von Windows XP zu Windows Vista

Sie müssen eine Verknüpfung zum primären Datencontainer eines Pakets erstellen. Sie können keine Verknüpfung zu anderen Verknüpfungspaketen erstellen.

Pfadnamen sind auf dem Arbeitscomputer, da die Verknüpfung sich während der Laufzeit auf den Clientcomputer auswirkt. Die verknüpften Pakete können mit einem Semikolon getrennt werden. Weitere Informationen zu Pfadnamenformaten finden Sie unter „Pfadnamenformate für Application Link“ auf Seite 86.

Beispiele

Wenn Sie das .NET Framework im Paket `dotnet.exe` verpacken und eine .NET-Anwendung haben, können Sie angeben, dass die Anwendung mit der `dotnet.exe`-Datei verknüpft werden muss, bevor sie gestartet werden kann.

```
RequiredAppLinks=c:\abs\path\dotnet.exe
```

Sie können einen relativen Pfad angeben.

```
RequiredAppLinks=<Relativer_Pfad>\dotnet.exe
```

Sie können einen UNC-Pfad angeben.

```
RequiredAppLinks=\\server\share\dotnet.exe
```

Sie können ThinApp-Ordnermakros im Pfad verwenden.

```
RequiredAppLinks=%SystemSystem%\Package.dat
```

Sie können Umgebungsvariablen im Pfad verwenden. Bei der Verwendung von Umgebungsvariablen besteht die Gefahr, dass ein Benutzer vor dem Start der Anwendung die Werte ändert und dadurch eine andere Application Link-Abhängigkeit erstellt, als der Administrator eingerichtet hat.

```
RequiredAppLinks=%MyEnvironmentVariable%\Package.dat
```

Sie können ein einzelnes Paket importieren, das in demselben Verzeichnis gespeichert ist wie die ausführbare Basisdatei.

```
RequiredAppLinks=Plugin.exe
```

Sie können ein einzelnes Paket importieren, das in demselben Unterverzeichnis gespeichert ist wie die ausführbare Basisdatei.

```
RequiredAppLinks=plugins\Plugin.exe
```

Sie können alle ausführbaren Dateien importieren, die in dem Verzeichnis für Plug-In-Dateien gespeichert sind. Wenn ThinApp eine ausführbare Datei nicht importieren kann, weil die Datei kein gültiges ThinApp-Paket ist oder weil ein Sicherheitsproblem vorliegt, schlägt das Laden der ausführbaren Datei fehl.

```
RequiredAppLinks=plugins\*.exe
```

Sie können alle ausführbaren Dateien importieren, die unter dem absoluten Pfad `n:\plugins` gespeichert sind.

```
RequiredAppLinks=n:\plugins\*.exe
```

Sie können die Umgebungsvariable `PLUGINS` erweitern und alle ausführbaren Dateien an diesem Speicherort importieren.

```
RequiredAppLinks=%PLUGINS%\*.exe
```

Sie können zwei angegebene Plug-In-Dateien laden und eine Liste mit ausführbaren Dateien, die am Speicherort für Plug-Ins gespeichert sind.

```
RequiredAppLinks=plugin1.exe;plugin2.exe;plugins\*.exe
```

OptionalAppLinks

Der Parameter `OptionalAppLinks` ist ähnlich wie der Parameter `RequiredAppLinks`, ignoriert jedoch Fehler und startet die Hauptanwendung auch dann, wenn ein Importvorgang fehlschlägt.

Sie müssen eine Verknüpfung zum primären Datencontainer eines Pakets erstellen. Sie können keine Verknüpfung zu anderen Verknüpfungspaketen erstellen.

Pfadnamen sind auf dem Arbeitscomputer, da die Verknüpfung sich während der Laufzeit auf den Clientcomputer auswirkt. Sie können absolute Pfade angeben, z. B. `c:\abs\path\dotnet.exe`, relative Pfade, z. B. `relpath\dotnet.exe`, und UNC-Pfade, z. B. `\\server\share\dotnet.exe`.

Die Parameter `RequiredAppLinks` und `OptionalAppLinks` verwenden dieselbe Syntax. Weitere Informationen zum Parameter `RequireAppLinks` sowie entsprechende Beispiele finden Sie unter „RequiredAppLinks“ auf Seite 87.

Konfigurieren von Anwendungs-Updates mit Application Sync

Mit dem Dienstprogramm Application Sync können bereitgestellte virtuelle Anwendungen auf dem neuesten Stand gehalten werden. Wenn eine Anwendung gestartet wird, kann Application Sync einen Webserver abfragen, um zu ermitteln, ob eine aktualisierte Version des Pakets verfügbar ist. Sofern ein Update verfügbar ist, lädt ThinApp die Unterschiede zwischen dem vorhandenen Paket und dem neuen Paket herunter und erstellt eine aktualisierte Version des Pakets.

Das Dienstprogramm Application Sync lädt Updates im Hintergrund herunter. Sie können weiterhin eine alte Version der Anwendung benutzen. Beendet der Benutzer die Anwendung vor Abschluss des Downloadvorgangs, so wird der Downloadvorgang fortgesetzt, sobald die virtuelle Anwendung erneut gestartet wird. Im Anschluss an den Downloadvorgang aktiviert ThinApp die neue Version beim nächsten Starten der Anwendung.

Sie müssen die Auskommentierung des Parameters AppSyncURL aufheben, um sämtliche Application Sync-Parameter zu aktivieren. Bei den folgenden Einträgen handelt es sich um Standardeinstellungen für Application Sync-Parameter:

```
AppSyncURL=https://example.com/some/path/PackageName.exe
AppSyncUpdateFrequency=1d
AppSyncExpirePeriod=30d
AppSyncWarningPeriod=5d
AppSyncWarningFrequency=1d
AppSyncWarningMessage=Diese Anwendung ist in AppSyncWarningPeriod Tagen nicht mehr für die
Benutzung verfügbar, wenn sie keine Verbindung zu ihrem Update-Server herstellen kann. Überprüfen
Sie Ihre Netzwerkverbindung, um den unterbrechungsfreien Dienst sicherzustellen.
AppSyncExpireMessage=Diese Anwendung konnte AppSyncExpirePeriod Tage lang keine Verbindung zu
ihrem Update-Server herstellen und ist daher nicht mehr für die Benutzung verfügbar. Überprüfen
Sie Ihre Netzwerkverbindung und versuchen Sie es erneut
AppSyncUpdatedMessage=
AppSyncClearSandboxOnUpdate=0
```

AppSyncClearSandboxOnUpdate

Der Parameter AppSyncClearSandboxOnUpdate gibt an, ob der Inhalt der Sandbox nach einem Update gelöscht werden soll.

ThinApp legt für den Parameter AppSyncClearSandboxOnUpdate einen Anfangswert fest, mit dem der gesamte Inhalt der Sandbox beibehalten wird.

Beispiele

Sie können den Parameter AppSyncClearSandboxOnUpdate so ändern, dass der Inhalt der Sandbox nach Updates für die Anwendungen gelöscht wird.

```
AppSyncClearSandboxOnUpdate=1
```

AppSyncExpireMessage

Der Parameter AppSyncExpireMessage legt fest, welche Meldung angezeigt wird, wenn die Verbindung mit dem Webserver nach der Ablaufperiode fehlschlägt und eine virtuelle Anwendung gestartet wird. Die Anwendung wird beendet, wenn die Meldung angezeigt wird.

Beispiele

ThinApp liefert eine Standardmeldung für den Parameter AppSyncExpireMessage.

```
AppSyncExpireMessage=Diese Anwendung konnte <AnwSyncAblaufZeit_Wert> Tage lang keine Verbindung
zu ihrem Update-Server herstellen und ist daher nicht mehr für die Benutzung verfügbar.
Überprüfen Sie Ihre Netzwerkverbindung und versuchen Sie es erneut. Überprüfen Sie Ihre
Netzwerkverbindung und versuchen Sie es erneut.
```

Ist der Wert des Parameters AppSyncExpirePeriod in Stunden oder Minuten angegeben, ändern Sie die Mitteilung, um Stunden oder Minuten anstatt Tage anzugeben.

AppSyncExpirePeriod

Der Parameter AppSyncExpirePeriod legt den Ablaufzeitpunkt für das Paket in Minuten (m), Stunden (h) oder Tagen (d) fest. Kann ThinApp den Webserver für die Suche nach Updates nicht erreichen, so funktioniert das Paket so lange weiter, bis die Ablaufperiode beendet ist und der Benutzer es schließt. Selbst nachdem die Ablaufperiode endet, versucht ThinApp den Webserver nach jedem folgenden Startversuch zu erreichen.

Beispiele

Sie können verhindern, dass das Paket abläuft, indem Sie den Standardwert never (nie) angeben.

```
AppSyncExpirePeriod=never (nie)
```

AppSyncURL

Der Parameter `AppSyncURL` legt die URL des Webservers oder den Speicherort der Dateifreigaben fest, auf bzw. an dem die aktualisierte Version einer Anwendung gespeichert wird. ThinApp überprüft diesen Speicherort und lädt das aktualisierte Paket herunter.

Application Sync funktioniert mit HTTP (unsicher), HTTPS (sicher) und Dateiprotokollen. Zum HTTPS-Protokoll gehört auch die Überprüfung der Identität des Webservers. Sie können einen Benutzernamen und ein Kennwort im Parameter `AppSyncURL` angeben, um eine Basisauthentifizierung zu gewährleisten. ThinApp übernimmt die Standardeinstellungen von Internet Explorer für den Proxyserver.

Sie müssen die Auskommentierung des Parameters `AppSyncURL` aufheben, um sämtliche Application Sync-Parameter zu aktivieren.

Beispiele

Sie können dem Parameter `AppSyncURL` einen HTTP- oder HTTPS-Wert im folgenden Format zuweisen.

```
AppSyncURL=https://<Website.com>/<Pfad>/<Paket_Name>.exe
```

Sie können Pfade zu lokalen Laufwerken und zu Netzlaufwerken angeben.

```
file:///C:/<Pfad>/<Paket_Name>.exe
```

Sie können einen UNC-Pfad angeben und auf die Speicherorte von Netzwerkressourcen zugreifen.

```
file://<Server>/<Freigabe>/<Pfad>/<Paket_Name>.exe
```

AppSyncUpdateFrequency

Der Parameter `AppSyncUpdateFrequency` gibt an, wie häufig ThinApp den Webserver auf Anwendungs-Updates überprüft. Sie können das Update-Intervall in Minuten (m), Stunden (h) oder Tagen (d) festlegen.

ThinApp legt als Anfangswert `1d` fest, damit ein Paket einmal täglich eine Verbindung zum Webserver herstellt, um nach Updates zu suchen. Ist dieselbe Sandbox für eine andere laufende Anwendung freigegeben, sucht ThinApp nicht nach einem Update.

Beispiele

Sie können für den Parameter `AppSyncUpdateFrequency` stattdessen den Wert auf 0 setzen, damit die Anwendung jedes Mal, wenn sie gestartet wird, nach Updates sucht.

```
AppSyncUpdateFrequency=0
```

AppSyncUpdatedMessage

Der Parameter `AppSyncUpdatedMessage` legt fest, welche Meldung beim ersten Starten eines aktualisierten Pakets angezeigt wird.

Beispiele

Sie können mithilfe des Parameters `AppSyncUpdatedMessage` überprüfen, ob eine Anwendung aktualisiert wurde.

```
AppSyncUpdatedMessage=Ihre Anwendung wurde aktualisiert.
```

AppSyncWarningFrequency

Der Parameter `AppSyncWarningFrequency` gibt an, wie häufig eine Warnmeldung angezeigt werden soll, bevor das Paket abläuft. Sie können den Zeitraum in Minuten (m), Stunden (h) oder Tagen (d) angeben.

ThinApp legt als Anfangswert `1d` fest, damit die Warnung einmal täglich angezeigt wird.

Beispiele

Sie können den Parameter `AppSyncWarningFrequency` so konfigurieren, dass die Warnung bei jedem Start der Anwendung angezeigt wird.

```
AppSyncWarningFrequency=0
```

AppSyncWarningMessage

Der Parameter `AppSyncWarningMessage` legt fest, welche Meldung angezeigt wird, wenn die Warnperiode beginnt. Beim ersten Starten der Anwendung in der Warnperiode wird eine Warnmeldung angezeigt und ThinApp versucht, über den Server auf die Updates zuzugreifen. Kann ThinApp das Paket nicht aktualisieren, so versucht ThinApp dies erneut bei jedem Starten der Anwendung. Die Warnmeldung wird nur nach dem Ablauf der einzelnen `AppSyncWarningFrequency`-Perioden angezeigt.

Beispiele

ThinApp enthält eine Standardmeldung für die Warnmeldung des Dienstprogramms Application Sync.

```
AppSyncWarningMessage=This application will become unavailable for use in %%remaining_days%%
day(s) if it cannot contact its update server. Check your network connection to ensure
uninterrupted service. (Diese Anwendung ist in %%remaining_days%% Tag(en) nicht mehr für die
Benutzung verfügbar, wenn sie keine Verbindung zu ihrem Update-Server herstellen kann. Überprüfen
Sie Ihre Netzwerkverbindung, um den unterbrechungsfreien Dienst sicherzustellen.)
```

Die Variable `%%remaining_days%%` gibt die Anzahl der Tage an, die noch bis zum Ablauf des Pakets verbleiben.

Ist der Wert des Parameters `AppSyncWarningPeriod` in Stunden oder Minuten angegeben, ändern Sie die Mitteilung, um Stunden oder Minuten anstatt Tage anzugeben.

AppSyncWarningPeriod

Der Parameter `AppSyncWarningPeriod` legt den Beginn der Warnperiode fest, bevor das Paket abläuft. Sie können den Zeitraum in Minuten (m), Stunden (h) oder Tagen (d) angeben. Wenn die Warnperiode beginnt, prüft ThinApp den Webserver jedes Mal wenn eine Anwendung gestartet wird und setzt den Wert des Parameters `AppSyncUpdateFrequency` auf 0.

Beispiele

Der Standardzeitraum des Parameters `AppSyncWarningPeriod` beträgt fünf Tage.

```
AppSyncWarningPeriod=5d
```

Konfigurieren von MSI-Dateien

Mithilfe von ThinApp-Parametern können Sie MSI-Dateien für die Bereitstellung durch Desktopmanagementsysteme konfigurieren.

Informationen über die Arbeit mit MSI-Dateien erhalten Sie unter „Erstellen einer MSI-Datenbank“ auf Seite 33.

Informationen zur Komprimierung von MSI-Dateien finden Sie im Zusammenhang mit anderen Parametern, die Dateigrößen steuern. Siehe „MSICompressionType“ auf Seite 79.

MSIArpProductIcon

Der Parameter `MSIArpProductIcon` gibt an, welche Symbole zur Darstellung der Anwendung im Windows-Dialogfeld **Software** verwendet werden sollen. Das Symbol kann in ICO- oder DLL-Dateien oder ausführbaren Dateien gespeichert sein.

Dieser Parameter darf nicht geändert werden. Wenn einem MSI-Paket kein Anwendungssymbol zugeordnet ist, wird die Anwendung mit einem allgemeinen Symbol angezeigt.

Beispiele

Der Parameter `MSIArpProductIcon` kann ein Symbol für Microsoft Office 2007 angeben. In diesem Beispiel verweist eine Indexnummer auf das erste Symbol in einer DLL-Datei.

```
MSIArpProductIcon=%Program Files Common%\Microsoft Shared\OFFICE12\
              Office Setup Controller\OSETUP.DLL,1
```

Der Eintrag `<Symbolindexnummer>` im Format `MSIArpProductIcon=<Pfad_zur_Symboldatei>[,<Symbolindexnummer>]` kann nur dann verwendet werden, wenn sich mehrere Symbole in einer DLL-Datei oder ausführbaren Datei befinden.

MSIDefaultInstallAllUsers

Der Parameter `MSIDefaultInstallAllUsers` legt den Installationsmodus der MSI-Datenbank fest. Sie können eine `.msi`-Datei für alle Benutzer auf einem Computer und für einzelne Benutzer installieren.

Informationen über die Erzwingung einer MSI-Installation für jeden Benutzer oder Rechner erhalten Sie unter „Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner“ auf Seite 34.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Beispiele

ThinApp legt für den Parameter `MSIDefaultInstallAllUsers` einen Anfangswert fest, mit dem die MSI-Datenbank mit Verknüpfungen und Dateitypzuordnungen für alle Benutzer, die sich auf dem Computer anmelden, installiert wird. Der Benutzer, der die Datenbank installiert, muss über Administratorberechtigungen verfügen. Mit dieser Vorgehensweise können Sie die Anwendung auf die Desktops aller Benutzer verschieben.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIDefaultInstallAllUsers=1
```

Ein einzelner Benutzer kann die MSI-Datenbank mit Verknüpfungen und Dateitypzuordnungen nur für diesen einen Benutzer installieren. Für die Installation eines einzelnen Benutzers sind keine Administratorrechte erforderlich. Gehen Sie auf diese Weise vor, wenn Sie möchten, dass jeder einzelne Benutzer die Anwendung separat benutzt.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIDefaultInstallAllUsers=0
```

Ein Administrator kann die MSI-Datenbank für alle Benutzer auf einem Rechner installieren, während ein einzelner Benutzer ohne Administratorberechtigungen die Datenbank nur für diesen einen Benutzer installieren kann.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIDefaultInstallAllUsers=2
```

MSIFilename

Der Parameter `MSIFilename` löst die Generierung einer MSI-Datenbank aus und gibt deren Dateinamen an. Andere MSI-Parameter können nur funktionieren, wenn Sie die Auskommentierung des Parameters `MSIFilename` aufheben.

Der Parameter erzeugt eine Windows Installer-Datei mit dem angegebenen Dateinamen im Ausgabeverzeichnis. Sie können eine MSI-Datei erstellen, wenn Sie Pakete über Desktopmanagementsysteme an Remote-Systeme liefern möchten. Anders als bei ausführbaren Dateien, bei denen das Dienstprogramm `thinreg.exe` manuell ausgeführt werden muss, ist die Erstellung von Verknüpfungen und Dateitypzuordnungen für jeden Benutzer bei MSI-Dateien automatisiert.

ThinApp kommentiert den Parameter `MSIFilename` so lange aus, bis Sie die Generierung der MSI-Datei während des Kapselungsprozesses angeben.

Beispiele

Der Bestandsname ist der Standardname im Parameter `MSIFilename`.

```
[BuildOptions]
;MSIFilename=<Bestandsname>.msi
```

Sie können während des Build-Prozesses eine MSI-Datei generieren und den Platzhalter „MSI-Dateiname“ durch Ihren eigenen Dateinamen ersetzen.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
```

MSIInstallDirectory

Der Parameter `MSIInstallDirectory` gibt den relativen Pfad des MSI-Installationsverzeichnisses an. Der Pfad ist bei Installationen auf jedem einzelnen Rechner relativ zum Verzeichnis `%ProgramFilesDir%` und bei Installationen für jeden einzelnen Benutzer relativ zum Verzeichnis `%AppData%`.

Wenn Sie die MSI-Datenbank für alle Benutzer installieren, platziert ThinApp während der Installation auf dem jeweiligen Rechner die Anwendungen im Verzeichnis `c:\%ProgramFilesDir%\<Bestandsname> (VMware ThinApp)`.

Wenn Sie die MSI-Datenbank für einzelne Benutzer installieren, platziert ThinApp die Anwendungen im Verzeichnis `c:\%AppData%\<Bestandsname> (VMware ThinApp)`.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Beispiele

Wenn Sie nicht möchten, dass der Parameter `MSIInstallDirectory` einen auf dem Bestandsnamen basierenden Speicherort verwendet, können Sie eine `.msi`-Datei im Verzeichnis `c:\Programme\<Eigene_Anw>` installieren.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIInstallDirectory=<Eigene_Anw>
```

MSIManufacturer

Der Parameter `MSIManufacturer` gibt den Hersteller oder das Paketierungsunternehmen der MSI-Datenbank an und zeigt den Wert im Windows-Dialogfeld **Software** an.

ThinApp legt als Anfangswert für den Parameter `MSIManufacturer` den Namen des Unternehmens fest, bei dem Ihre Windows-Lizenz registriert ist.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Beispiele

Sie können den Parameter `MSIManufacturer` so ändern, dass der Name einer bestimmten Abteilung angezeigt wird. Dann sehen die Benutzer im Windows-Dialogfeld **Software** die Abteilung und können sich beispielsweise an das Helpdesk für diese Abteilung wenden.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIManufacturer=<Name_der_Abteilung_oder_des_Unternehmens>
```

MSIProductCode

Der Parameter `MSIProductCode` gibt einen Produktcode für die MSI-Datenbank an. Windows Installer verwendet den Code zur Identifizierung von MSI-Paketen.

Der Kapselungsprozess generiert einen zufälligen und eindeutigen Produktcode, der nicht aus der Anwendung abgerufen wird. Bei dem Wert muss es sich um eine gültige GUID (Globally Unique Identifier) handeln.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Der Parameter `MSIProductCode` darf nicht geändert werden.

Beispiele

Der Kapselungsprozess kann eine `MSI`-Datei mit dem Produktcode 590810CE-65E6-3E0B-08EF-9CCF8AE20D0E erstellen.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIProductCode={590810CE-65E6-3E0B-08EF-9CCF8AE20D0E}
```

MSIProductVersion

Der Parameter `MSIProductVersion` gibt eine Produktversionsnummer für die MSI-Datenbank an, um die Versionskontrolle zu ermöglichen. Diese Versionsnummer ist von der Anwendungsversion oder ThinApp-Version vollkommen unabhängig.

ThinApp weist die anfängliche Versionsnummer 1.0 zu. Diese Versionsnummer wird in den Eigenschaften der Datenbank angezeigt.

Wenn Sie ein Paket auf einem Rechner installieren, auf dem dasselbe Paket bereits installiert ist, überprüft Windows Installer die Versionsnummern und sperrt die Installation einer älteren Version gegenüber einer aktuelleren Version. Unter diesen Umständen müssen Sie die neue Version deinstallieren.

Der Parameter `MSIProductVersion` funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Beispiele

Sie können den Wert des Parameters `MSIProductVersion` ändern, wenn Sie Änderungen am MSI-Paket vornehmen. Wenn Sie den Wert 2.0 angeben, deinstalliert ThinApp die Version 1.0 des Pakets und installiert anschließend die Version 2.0 des Pakets.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIProductVersion=2.0
```

Der Wert des Parameters `MSIProductVersion` hat folgendes Format: `X.Y.Z`. Die Werte für `X` und `Y` können zwischen 0 und 255 liegen, während der Wert für `Z` zwischen 0 und 65536 liegen kann.

MSIRequireElevatedPrivileges

Der Parameter `MSIRequireElevatedPrivileges` gilt für Windows Vista und gibt die Anforderung erhöhter Rechte für die MSI-Datenbank an.

Die meisten Benutzer, die sich bei Windows Vista anmelden, haben eingeschränkte Rechte. Um MSI-Pakete für alle Benutzer zu installieren, die Verknüpfungen und Dateitypzuordnungen benötigen, müssen die Benutzer erhöhte Rechte haben.

ThinApp legt für den Parameter `MSIRequireElevatedPrivileges` einen Anfangswert fest, der die MSI-Datenbank mit der Kennzeichnung versieht, dass sie erhöhte Rechte erfordert. Wenn Ihr System für UAC-Eingabeaufforderungen konfiguriert ist, wird eine UAC-Eingabeaufforderung angezeigt, wenn Sie eine Anwendung installieren.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Beispiele

Sie können den Parameter `MSIRequireElevatedPrivileges` so ändern, dass die UAC-Eingabeaufforderung und die Installation auf allen Computern blockiert wird.

```
[BuildOptions]
MSIFilename=<MSI-Dateiname>.msi
MSIRequireElevatedPrivileges=0
```

MSIUpgradeCode

Der Parameter `MSIUpgradeCode` gibt einen Code für die MSI-Datenbank an, der Updates ermöglicht. Wenn zwei Pakete, z. B. Version 1.0 und Version 2.0 eines Pakets, denselben Upgrade-Code haben, erkennt das MSI-Installationsprogramm diese Verknüpfung, deinstalliert die niedrigere Version und installiert anschließend das aktualisierte Paket.

Der Kapselungsprozess generiert basierend auf dem Bestandsnamen einen zufälligen Upgrade-Code. Um sicherzustellen, dass die MSI-Datenbankversionen denselben Upgrade-Code haben, sollte der Bestandsname in allen Versionen des MSI-Wrappers identisch sein. Weitere Informationen zum Bestandsnamen finden Sie unter „InventoryName“ auf Seite 96.

Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Windows Installer-Datenbank anfordert.

Der Wert des Parameters `UpgradeCode` darf nur geändert werden, wenn der neue Wert eine gültige GUID ist.

Beispiele

Der Kapselungsprozess kann eine MSI-Datei mit dem Upgrade-Code D89F1994-A24B-3E11-0C94-7FD1E13AB93F erstellen.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIUpgradeCode={D89F1994-A24B-3E11-0C94-7FD1E13AB93F}
```

MSIUseCabs

Der Parameter `MSIUseCabs` legt die Verwendung von `.cab`-Dateien fest, was Auswirkungen auf die Anwendungsleistung haben kann.

ThinApp legt einen Anfangswert fest, mit dem die Paketdateien in einer `.cab`-Datei komprimiert werden und das Verschieben der Datei erleichtert wird. Die `.cab`-Datei befindet sich in der MSI-Datei.

Beispiele

Sie können den Parameter `MSIUseCabs` so ändern, dass keine `.cab`-Datei verwendet wird, wenn dies den Installationsprozess für Anwendungen verlangsamen würde. Sie können die MSI-Datei und die einzelnen ausführbaren Dateien im Verzeichnis `/bin` verteilen, um die Anwendung zu installieren.

```
[BuildOptions]
MSIUseCabs=0
```

Konfigurieren von Sandbox-Speicher und Bestandsnamen

Mithilfe von ThinApp-Parametern können Sie die Sandbox konfigurieren, in der alle von der gekapselten Anwendung vorgenommenen Änderungen gespeichert werden. Der ThinApp-Bestandsname kann bewirken, dass der Sandboxname geändert werden muss.

Weitere Informationen zur Platzierung und Struktur der Sandbox finden Sie unter Kapitel 6, „Suche nach der ThinApp-Sandbox“, auf Seite 99.

InventoryName

Der Parameter `InventoryName` ist eine Zeichenfolge, die Dienstprogramme für die Bestandsnachverfolgung zur Paketidentifizierung verwenden. Dieser Parameter bestimmt während der Kapselung der Anwendung die Standardnamen des Projektordners und der Sandbox.

Bei der Kapselung der Anwendung wird ein Standardwert für den Parameter `InventoryName` festgelegt. Dies erfolgt basierend auf neuen Zeichenfolgen, die unter einem der folgenden Speicherorte erstellt wurden:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`

Das Dienstprogramm `thinreg.exe` und die MSI-Dateien von ThinApp referenzieren diesen Parameter, um den Produktnamen zu ermitteln, der unter „Software“ in der Systemsteuerung angezeigt werden soll. Wenn der Bestandsname beispielsweise `SuperApp` lautet und Sie mit dem Dienstprogramm `thinreg.exe` eine MSI-Datei installieren oder ein Paket registrieren, wird in der Liste „Software“ eine installierte Anwendung mit der Zeichenfolge `SuperApp (VMware ThinApp)` angezeigt. ThinApp hängt `VMware ThinApp` an den Bestandsnamen an, um Anwendungen zu unterscheiden, die bei Bestandsüberprüfungen virtualisiert wurden.

Sie können denselben Bestandsnamen über verschiedene Versionen derselben Anwendung hinweg verwenden, um sicherzustellen, dass nur die neueste Version in der Liste „Software“ angezeigt wird. Die Anwendungen in der Liste „Software“ überschreiben einander und verhindern, dass alle registrierten Pakete deinstalliert werden. Möchten Sie mehr als eine Version deinstallieren, müssen Sie für jede Version einen eigenen Bestandsnamen verwenden. Verwenden Sie beispielsweise Microsoft Office 2003 und Microsoft Office 2007 als Bestandsnamen anstatt nur Microsoft Office. Wenn Sie verschiedene Versionen einer virtuellen Anwendung in derselben Umgebung pflegen, sollten Sie möglicherweise den Parameter `SandboxName` ändern, um sicherzustellen, dass eine neue Version die Benutzereinstellungen in einer anderen Sandbox isoliert hat.

Wenn Sie ein Paket haben, das andere Anwendungen einschließt, müssen Sie möglicherweise den Bestandsnamen manuell aktualisieren, damit der wirkliche Inhalt des Pakets wiedergegeben wird. Wenn Sie beispielsweise die Anwendung `SuperApp` kapseln und das Paket Java Runtime einschließt, kann als Wert für `InventoryName` `Java Runtime Environment 1.5` anstelle von `SuperApp` angezeigt werden. Die Liste „Software“ zeigt die erste im Paket installierte Anwendung an.

Beispiele

Sie können für den Parameter `InventoryName` den Wert „Microsoft Office 2003“ angeben.

```
[BuildOptions]
InventoryName=Microsoft Office 2003
```

RemoveSandboxOnExit

Der Parameter `RemoveSandboxOnExit` löscht die Sandbox und setzt die Anwendung zurück, wenn der letzte untergeordnete Prozess beendet wird.

ThinApp speichert alle von der Anwendung in der Registrierung und an den Speicherorten des Dateisystems vorgenommenen Änderungen mit dem Isolationsmodus „WriteCopy“ oder „Full“ in der Sandbox. ThinApp legt für den Parameter `RemoveSandboxOnExit` einen Anfangswert fest, der dafür sorgt, dass auch bei mehrfachem Ausführen der Anwendung die Einstellungen für das Sandbox-Verzeichnis konsistent bleiben.

Wenn die Anwendung untergeordnete Prozesse erstellt, löscht ThinApp die Sandbox erst, nachdem alle untergeordneten Prozesse beendet wurden. Anwendungen können so ausgelegt sein, dass die untergeordneten Prozesse bestehen bleiben und die Bereinigung darum blockieren. Zum Beispiel behält Microsoft Office 2003 den Prozess `ctfmon.exe` bei. Sie können den Prozess `ctfmon.exe` und die untergeordneten Prozesse mithilfe eines Skripts beenden, um das Ausführen der Bereinigung zu erzwingen.

Sie können während der Laufzeit entscheiden, ob die API-Funktion des Skripts `RemoveSandboxOnExit` zum Löschen der Sandbox bei Beenden verwendet werden soll.

Beispiele

Sie können den Parameter `RemoveSandboxOnExit` so ändern, dass die Sandbox bei Beendigung der Anwendung gelöscht wird. Wenn eine Anwendung von mehreren Benutzern unter demselben Benutzernamen gemeinsam genutzt wird, können Sie die Sandbox löschen, um die Änderungen, die der vorherige Benutzer in der Registrierung und am Dateisystem vorgenommen hat, zu entfernen.

```
[BuildOptions]
RemoveSandboxOnExit=1
```

SandboxName

Der Parameter `SandboxName` gibt den Namen des Verzeichnisses an, in dem die Sandbox gespeichert wird.

Mit dem von ThinApp festgelegten Anfangswert wird der Bestandsname als Sandboxname verwendet.

Beim Upgrade einer Anwendung können Sie mithilfe des Sandbox-Namens ermitteln, ob Benutzer die vorherigen persönlichen Einstellungen beibehalten oder neue Einstellungen benötigen. Eine Änderung des Sandbox-Namens bei neuen Installationen wirkt sich auf die Notwendigkeit zur Erstellung einer neuen Sandbox mit anderen Einstellungen oder zur Beibehaltung derselben Sandbox aus.

Beispiele

Wenn Sie eine Anwendung aktualisieren und neue Benutzereinstellungen für die Anwendung verwenden möchten, können Sie den Parameter `SandboxName` ändern, um die aktualisierte Version kenntlich zu machen.

```
[BuildOptions]
SandboxName=Meine Anwendung 2.0
```

SandboxNetworkDrives

Der Parameter `SandboxNetworkDrives` legt fest, ob ThinApp Schreibvorgänge unabhängig von den Isolationsmoduseinstellungen auf ein Netzlaufwerk oder in die Sandbox umleitet.

Wenn Sie mithilfe dieses Parameters Schreibvorgänge auf Netzlaufwerke umleiten oder die Einstellung des Isolationsmodus für das Laufwerk auf „Merged“ setzen, führt dies in beiden Fällen zu demselben Ergebnis.

Beispiele

Wenn Sie Speicherplatz sparen möchten oder Dateien für die Zusammenarbeit freigeben möchten, behalten Sie die Standardeinstellung des Parameters `SandboxNetworkDrives` bei, damit Schreibvorgänge ohne Änderungen in der Sandbox zu speichern auf Netzlaufwerke umgeleitet werden.

```
[BuildOptions]
SandboxNetworkDrives=0
```

Sie können Änderungen in der Sandbox speichern und verhindern, dass die Benutzer Änderungen an Netzlaufwerken vornehmen.

```
[BuildOptions]
SandboxNetworkDrives=1
```

SandboxPath

Der Parameter `SandboxPath` legt den Pfad der Sandbox fest.

Der Pfad der Sandbox kann relativ oder absolut angegeben werden, Ordnermakros oder Umgebungsvariablen enthalten und sich auf einem Netzlaufwerk befinden. Weitere Informationen zur Festlegung des anfänglichen Speicherorts für die Sandbox oder zur Suche nach der Sandbox durch ThinApp finden Sie unter „Suchreihenfolge für die Sandbox“ auf Seite 99.

Sie können den Parameter `SandboxPath` für die Erfordernisse im Hinblick auf lokale Laufwerke, USB-Laufwerke oder Netzlaufwerke, auf Größenbeschränkungen des anfänglichen Sandboxspeicherorts oder auf die Verschiebung der Sandbox auf den Desktop zu Fehlerbehebungszwecken entsprechend anpassen.

Der Name der Sandbox ist im Pfad der Sandbox nicht enthalten, weil dieser separat im Parameter `SandboxName` festgelegt wird.

Beispiele

Sie können den Parameter `SandboxPath` so ändern, dass die Sandbox in demselben Verzeichnis erstellt wird wie die ausführbare Datei. Wenn `Mozilla Firefox 3.0` der Wert des Parameters `SandboxName` ist, können Sie die Sandbox `Mozilla Firefox 3.0` in demselben Verzeichnis erstellen, von dem aus Firefox ausgeführt wird.

```
[BuildOptions]
SandboxPath=.
```

Sie können die Sandbox in einem Unterverzeichnis erstellen, das dem Speicherort der ausführbaren Datei untergeordnet ist.

```
[BuildOptions]
SandboxPath=LocalSandbox\Subdir1
```

Sie können die Sandbox im Ordner `%AppData%` des jeweiligen Benutzers im Verzeichnis `Thinstall` erstellen.

```
[BuildOptions]
SandboxPath=%AppData%\Thinstall
```

Sie können die Sandbox auf einem zugeordneten Laufwerk speichern, um eine Sicherung der Sandbox zu erstellen oder um die Anwendungseinstellungen für Benutzer beizubehalten, die sich auf einem beliebigen Rechner anmelden. Wenn `Mozilla Firefox 3.0` der Wert des Parameters `SandboxName` ist, können Sie die Sandbox im Verzeichnis `z:\Sandbox\Mozilla Firefox 3.0` erstellen.

```
[BuildOptions]
SandboxPath=Z:\Sandbox
```

SandboxRemovableDisk

Der Parameter `SandboxRemovableDisk` bestimmt, ob die Anwendung Änderungen der Wechseldatenträger auf den Datenträgern oder in der Sandbox speichern kann. Wechseldatenträger sind zum Beispiel USB-Flash-Geräte und externe Festplatten.

Mit dem vom ThinApp festgelegten Anfangswert wird die Anwendung angewiesen, Änderungen an Dateien auf Wechseldatenträgern auf den Datenträgern zu speichern.

Beispiele

Um Speicherplatz zu sparen, können Sie den Parameter `SandboxRemovableDisk` so ändern, dass Änderungen der Wechseldatenträger in die Sandbox umgeleitet werden. Abhängig vom Isolationsmodus des Wechseldatenträgers können sich Änderungen an Dateien, die auf Wechseldatenträgern gespeichert sind, in der Sandbox oder auf dem Wechseldatenträger befinden.

```
[BuildOptions]
SandboxRemovableDisk=1
```

Suche nach der ThinApp-Sandbox 6

Die Sandbox ist das Verzeichnis, in dem alle Änderungen, die eine gekapselte Anwendung vornimmt, gespeichert werden. Sobald Sie die Anwendung das nächste Mal starten, werden diese Änderungen aus der Sandbox integriert. Wenn Sie das Sandbox-Verzeichnis löschen, wird die Anwendung auf den gekapselten Status zurückgesetzt.

Dieser Abschnitt umfasst die folgenden Themen:

- „Suchreihenfolge für die Sandbox“ auf Seite 99
- „Steuern des Sandbox-Speicherorts“ auf Seite 101
- „Sandbox-Struktur“ auf Seite 102

Suchreihenfolge für die Sandbox

Beim Starten der gekapselten Anwendung sucht ThinApp an bestimmten Speicherorten und in einer bestimmten Reihenfolge nach einer bereits vorhandenen Sandbox. ThinApp verwendet die Sandbox, die als erste erkannt wird. Kann ThinApp keine vorhandene Sandbox ermitteln, erstellt ThinApp eine Sandbox gemäß bestimmten Umgebungsvariablen und Parametereinstellungen. Überprüfen Sie die Suchreihenfolge und Erstellungslogik der Sandbox, bevor Sie die Platzierung der Sandbox verändern.

In dieser Suchreihenfolge wird Mozilla Firefox 3.0 als Beispiel mit folgenden Variablen verwendet:

- `<Sandbox_Name>` ist Mozilla Firefox 3.0

  Der `SandboxName`-Parameter in der `Package.ini`-Datei legt den Namen fest. Siehe „SandboxName“ auf Seite 97.

- `<Sandbox_Pfad>` ist `Z:\sandboxes`

  Der `SandboxPath`-Parameter in der `Package.ini`-Datei legt den Pfad fest. Siehe „SandboxPath“ auf Seite 98.

- `<exe_Verzeichnis>` ist `C:\Programme\Firefox`

  Die Anwendung startet von diesem Speicherort aus.

- `<computer_name>` ist MAXMUSTERMANN-COMPUTER

- `%AppData%` ist `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten`

  ThinApp fordert vom Betriebssystem den Ordnerspeicherort `Anwendungsdaten`. Der Speicherort hängt vom Betriebssystem oder der Konfiguration ab.

ThinApp startet die Sandbox-Suche, indem in dieser Reihenfolge nach folgenden Umgebungsvariablen gesucht wird:

- %<Sandbox_Name>_SANDBOX_DIR%

  Diese Umgebungsvariable ändert den Speicherort der Sandbox für bestimmte Anwendungen auf dem Computer. Ist zum Beispiel die Umgebungsvariable `Mozilla Firefox 3.0_SANDBOX_DIR` vorhanden, bestimmt ihr Wert den Speicherort des übergeordneten Verzeichnisses der Sandbox. Ist der Wert `z:\FirefoxSandbox`, bevor Sie die Anwendung starten, speichert ThinApp die Sandbox in `z:\FirefoxSandbox.MAXMUSTERMANN-COMPUTER`, wenn das Verzeichnis bereits besteht. Ist das Verzeichnis nicht vorhanden, erstellt ThinApp eine Sandbox in `z:\FirefoxSandbox`.

- %THINSTALL_SANDBOX_DIR%

  Diese Umgebungsvariable ändert den Speicherort jeder Sandbox auf einem Computer. Ist zum Beispiel die Umgebungsvariable `THINSTALL_SANDBOX_DIR` vorhanden, bestimmt ihr Wert den Speicherort des übergeordneten Verzeichnisses der Sandbox. Ist der Wert `z:\MySandboxes`, bevor Sie die Anwendung starten, erstellt ThinApp eine Sandbox in `z:\MySandboxes`.

Wenn ThinApp die Umgebungsvariable %<Sandbox_Name>_SANDBOX_DIR% oder %THINSTALL_SANDBOX_DIR% nicht erkennt, überprüft ThinApp die folgenden Dateisystemverzeichnisse und erstellt eine Sandbox im Verzeichnis, das als erstes erkannt wird:

- <exe_Verzeichnis>\<Sandbox_Name>.<computer_name>

  Zum Beispiel `C:\Programme\Firefox\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <exe_Verzeichnis>\<Sandbox_Name>

  Zum Beispiel `C:\Programme\Firefox\Mozilla Firefox 3.0`

- <exe_Verzeichnis>\Thinstall\<Sandbox_Name>.<computer_name>

  Zum Beispiel `C:\Programme\Firefox\Thinstall\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <exe_Verzeichnis>\Thinstall\<Sandbox_Name>

  Zum Beispiel `C:\Programme\Firefox\Thinstall\Mozilla Firefox 3.0`

- <Sandbox_Pfad>\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `Z:\sandboxes\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <Sandbox_Pfad>\<Sandbox_Name>

  Zum Beispiel `Z:\sandboxes\Mozilla Firefox 3.0`

- %AppData%\Thinstall\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten\Thinstall\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- %AppData%\Thinstall\<Sandbox_Name>

  Zum Beispiel `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten\Thinstall\Mozilla Firefox 3.0`

Wenn ThinApp die Umgebungsvariable %<Sandbox_Name>_SANDBOX_DIR% oder %THINSTALL_SANDBOX_DIR% und die angegebenen Dateisystemverzeichnisse nicht erkennt, erstellt ThinApp eine Sandbox mit folgenden Richtlinien in dieser Reihenfolge:

- Ist der Parameter `SANDBOXPATH` `Package.ini` festgelegt, bestimmt dieser Wert den Sandbox-Speicherort.
- Schließt ThinApp die Sandbox-Suche ergebnislos ab, erstellt ThinApp eine Sandbox im Standardverzeichnis `%AppData%\Thinstall` des Benutzers.

ANMERKUNG Es kann nur jeweils ein Computer eine freigegebene Sandbox verwenden. Verwendet ein Computer bereits eine Sandbox, erstellt ThinApp eine neue Sandbox, damit Sie mit Ihrer Arbeit fortfahren können, bis die vorherige Kopie der Sandbox geschlossen wird.

Steuern des Sandbox-Speicherorts

Der Setup Capture-Prozess fügt den Parameter `SandboxName` zur `Package.ini`-Datei hinzu. Wenn Sie Firefox kapseln und `Mozilla Firefox 3.0` der Wert dieses Parameters ist, lautet der Standardspeicherort der Sandbox für die Anwendung `%AppData%\Thinstall\Mozilla Firefox 3.0`. Der typische `%AppData%`-Speicherort ist `C:\Dokumente und Einstellungen\<Benutzer_Name>\Anwendungsdaten`. `%AppData%` wird oft einem freigegebenen Netzlaufwerk zugeordnet.

Speichern der Sandbox im Netzwerk

Sie können den `SandboxPath`-Parameter verwenden, um die Sandbox auf einem zugeordneten Laufwerk zu speichern. Ein Netzwerkspeicherort ist nützlich, um die Sandbox zu sichern und ebenso für Benutzer, die sich an einem beliebigen Computer anmelden und ihre Anwendungseinstellungen beibehalten möchten. Weitere Informationen über den `SandboxPath`-Parameter finden Sie unter „SandboxPath“ auf Seite 98.

So speichern Sie die Sandbox auf einem zugeordneten Laufwerk

1 Öffnen Sie die `Package.ini`-Datei.

2 Legen Sie unter dem `SandboxName`-Parameter den `SandboxPath`-Parameter auf den Netzwerkspeicherort fest.

```
SandboxName=Mozilla Firefox 3.0
SandboxPath=Z:\Sandbox
```

Ist zum Beispiel `Mozilla Firefox 3.0` der Wert des `SandboxName`-Parameters, erstellt die gekapselte Firefox-Anwendung die Sandbox unter `Z:\Sandbox\Mozilla Firefox 3.0`.

Speichern der Sandbox auf einem portablen Gerät

Sie können den `SandboxPath`-Parameter verwenden, um einen Sandbox-Speicherort auf einem portablen Gerät festzulegen. Sie können jedes portable Gerät wie zum Beispiel ein USB-Laufwerk verwenden, das als Festplattenlaufwerk im Systemordner `Arbeitsplatz` angezeigt wird. Ein Speicherort auf einem portablen Gerät ist nützlich, um die Sandbox-Daten auf dem Gerät zu speichern, auf dem sich die Anwendung befindet.

Weitere Informationen über den `SandboxPath`-Parameter finden Sie unter „SandboxPath“ auf Seite 98.

So speichern Sie die Sandbox auf einem USB-Laufwerk im selben Verzeichnis wie die ausführbare Datei

1 Öffnen Sie die `Package.ini`-Datei.

2 Legen Sie unter dem `SandboxName`-Parameter den `SandboxPath`-Parameter auf diesen Wert fest.

```
SandboxName=Mozilla Firefox 3.0
SandboxPath=.
```

Ist zum Beispiel `Mozilla Firefox 3.0` der Wert des `SandboxName`-Parameters, erstellt die gekapselte Firefox-Anwendung die Sandbox `Mozilla Firefox 3.0` im gleichen Verzeichnis, von dem aus Firefox ausgeführt wird.

So speichern Sie die Sandbox in einem Thinstall-Verzeichnis auf einem USB-Laufwerk auf derselben Ebene wie die ausführbare Datei

1 Ist die Umgebungsvariable `%THINSTALL_SANDBOX_DIR%` oder `%<Sandbox_Name>_SANDBOX_DIR%` festgelegt, deaktivieren Sie die Variablen.

2 Erstellen Sie ein `Thinstall`-Verzeichnis auf dem portablen Gerät in dem Verzeichnis, in dem sich Ihre gekapselte Anwendung befindet.

Wenn die paketierte Anwendung das nächste Mal von einem portablen Gerät aus gestartet wird, erstellt sie eine Sandbox im `Thinstall`-Verzeichnis.

3 Wurden die Anwendung und die Sandbox ursprünglich von einem anderen Speicherort, zum Beispiel einem Computer, gestartet und Sie benötigen dieselbe Sandbox auf einem portablen Gerät, kopieren Sie das `Thinstall`-Verzeichnis von `%AppData%` auf das Verzeichnis, in dem sich die ausführbare Datei auf dem Gerät befindet.

ThinApp verwendet die Sandbox am ursprünglichen Speicherort nicht mehr.

Sandbox-Struktur

ThinApp verwendet zum Speichern der Sandbox fast dieselbe Dateistruktur wie zum Erstellen des Projekts. ThinApp verwendet Makronamen für Speicherorte von Shell-Ordnern wie beispielsweise `%AppData%`, anstatt hartcodierter Pfade. Mithilfe dieser Struktur kann die Sandbox dynamisch in verschiedene Computer migriert werden, wenn die Anwendung von neuen Speicherorten gestartet wird.

Die Sandbox enthält die folgenden Registrierungsdateien:

- `Registry.rw.tvr` – Enthält alle Registrierungsänderungen, die die Anwendung vornimmt.
- `Registry.rw.lck` – Verhindert, dass andere Computer gleichzeitig eine auf einer Netzwerkfreigabe gespeicherte Registrierung verwenden.
- `Registry.tvr.backup` – Enthält eine Sicherungsdatei der `.tvr`-Datei, die ThinApp verwendet, wenn die ursprüngliche `.tvr`-Datei beschädigt ist.

Neben diesen Registrierungsdateien enthält die Sandbox Verzeichnisse, zu denen `%AppData%`, `%ProgramFilesDir%` und `%SystemRoot%` gehören. Jeder dieser Ordner enthält Änderungen der entsprechenden Ordner in der gekapselten Anwendung.

Änderungen an der Sandbox

ThinApp speichert Dateisysteminformationen in der virtuellen Registrierung. Die virtuelle Registrierung ermöglicht es ThinApp, den Dateisystemzugriff in der virtuellen Umgebung zu optimieren. Versucht zum Beispiel eine Anwendung eine Datei zu öffnen, muss ThinApp nicht im realen Dateisystem nach dem realen Speicherort und noch einmal nach dem Speicherort der Sandbox suchen. Stattdessen kann ThinApp alleine anhand der virtuellen Registrierung prüfen, ob die Datei vorhanden ist. Dadurch erhöht sich die Laufzeitleistung von ThinApp.

VMware bietet keine Unterstützung für das direkte Ändern oder Hinzufügen von Dateien zur Sandbox. Wenn Sie Dateien in das Sandbox-Verzeichnis kopieren, sind die Dateien für die Anwendung nicht sichtbar. Ist die Datei bereits in der Sandbox vorhanden, können Sie sie überschreiben und aktualisieren. VMware empfiehlt, dass Sie sämtliche Änderungen aus der Anwendung selbst vornehmen.

Auflisten virtueller Registrierungsinhalte mit vregtool

Da die Sandbox die Änderungen der Registrierung enthält, benötigen Sie möglicherweise das Dienstprogramm `vregtool`, um virtuelle Registrierungsänderungen anzuzeigen. Sie müssen über Zugriff auf das Dienstprogramm `vregtool` unter `C:\Programme\VMware\VMware ThinApp` verfügen.

Ein Beispielbefehl zur Auflistung der Inhalte einer virtuellen Registrierungsdatei ist `vregtool registry.rw.tvr printkeys`.

Erstellen von ThinApp-Snapshots und -Projekten von der Befehlszeile 7

Das Dienstprogramm `snapshot.exe` erstellt einen Snapshot eines Computerdateisystems und einer Registrierung und erstellt von zwei zuvor gekapselten Snapshots ein ThinApp-Projekt. Sie müssen das Dienstprogramm `snapshot.exe` nicht direkt starten, da es vom Setup Capture-Assistenten gestartet wird. Nur fortgeschrittene Benutzer und Systemintegratoren, die ThinAppOnly-Funktionalität in andere Plattformen integrieren, sollten dieses Dienstprogramm direkt verwenden.

Das Erstellen eines Snapshots von einem Computerdateisystem und einer Registrierung umfasst das Überprüfen und Speichern einer Kopie der folgenden Daten:

- Dateiinformationen für alle lokalen Laufwerke

  Diese Informationen umfassen Verzeichnisse, Dateinamen, Dateiattribute, Dateigrößen und Dateiänderungsdaten.

- Die Registrierungsstrukturen `HKEY_LOCAL_MACHINE` und `HKEY_USERS`

  ThinApp überprüft nicht die Registrierungseinträge `HKEY_CLASSES_ROOT` und `HKEY_CURRENT_USER` , weil diese Einträge Untereinträge der Einträge `HKEY_LOCAL_MACHINE` und `HKEY_USERS` sind.

Die `snapshot.ini`-Konfigurationsdatei gibt an, welche Verzeichnisse und Unterschlüssel von einem ThinApp-Projekt auszuschließen sind, wenn eine Anwendung gekapselt wird. Sie können diese Datei für bestimmte Anwendungen anpassen.

Dieser Abschnitt umfasst die folgenden Themen:

- „Methoden zur Verwendung des Dienstprogramms snapshot.exe“ auf Seite 103
- „Beispiele für snapshot.exe-Befehle“ auf Seite 105
- „Erstellen eines Projekts ohne den Setup Capture-Assistenten“ auf Seite 105
- „Anpassen der Snapshot.ini-Datei“ auf Seite 106

Methoden zur Verwendung des Dienstprogramms snapshot.exe

Mit dem Dienstprogramm `snapshot.exe` können Sie Snapshot-Dateien von Computerstatus, die Vorlagendatei für die `Package.ini`-Datei oder ein ThinApp-Projekt erstellen und den Inhalt einer Snapshot-Datei anzeigen.

Informationen über den vollständigen Vorgang zum Erstellen eines ThinApp-Projekts von der Befehlszeile finden Sie unter „Erstellen eines Projekts ohne den Setup Capture-Assistenten“ auf Seite 105.

Erstellen von Snapshots des Computerstatus

Das Dienstprogramm `snapshot.exe` erstellt eine Snapshot-Datei des Computerzustandes. ThinApp kapselt den Computerzustand und speichert ihn in einer einzelnen Datei, um ein Projekt zu erstellen. Das Dienstprogramm `snapshot.exe` speichert eine Kopie von Registrierungsdaten und Dateisystem-Metadaten, die Pfade, Dateinamen, Größen, Attribute und Zeitstempel umfassen.

Verwendung

```
snapshot.exe SnapshotFileName.snapshot [–Config ConfigFile.ini][BaseDir1][BaseDir2][BaseReg1]
```

Beispiele

```
Snapshot My.snapshot
Snapshot My.snapshot –Config MyExclusions.ini
Snapshot My.snapshot C:\MyAppDirectory HKEY_LOCAL_MACHINE\Software\MyApp
```

Optionen

Die Optionen geben die Verzeichnisse oder Unterschlüssel im Snapshot an.

Option	Beschreibung
`–Config ConfigFile.ini`	Gibt Verzeichnisse oder Registrierungsunterschlüssel an, die bei der Erstellung des Snapshots ausgeschlossen werden sollen. Wenn Sie keine Konfigurationsdatei angeben, verwendet ThinApp die `snapshot.ini`-Datei aus dem ThinApp-Installationsverzeichnis.
`BaseDir1`	Gibt eines oder mehrere Basisverzeichnisse an, die in die Überprüfung aufgenommen werden sollen. Wenn Sie keine Basisverzeichnisse angeben, überprüft das Dienstprogramm `snapshot.exe` `C:\` und alle Unterverzeichnisse. Wenn Sie einen Computer überprüfen, auf dem Windows oder Programmdateien auf verschiedenen Festplatten installiert sind, schließen Sie diese Laufwerke in Ihre Überprüfung ein. Wenn Sie wissen, dass Ihre Anwendungsinstallation Dateien an festen Speicherorten erstellt oder ändert, geben Sie diese Verzeichnisse an, um die Gesamtzeitdauer für das Überprüfen des Computers zu verringern.
`BaseReg1`	Gibt einen oder mehrere Registrierungsunterschlüssel an, die in die Überprüfung aufgenommen werden sollen. Wenn Sie keine Registrierungsunterschlüssel angeben, überprüft das Dienstprogramm `snapshot.exe` die Schlüssel `HKEY_LOCAL_MACHINE` und `HKEY_USERS`.

Erstellen der Template Package.ini-Datei aus zwei Snapshot-Dateien

Das Dienstprogramm `snapshot.exe` generiert eine `Package.ini`-Vorlagendatei. Das Dienstprogramm überprüft die beiden Snapshot-Dateien auf alle Anwendungen, die erstellt und auf die von Verknüpfungslinks oder dem Menü **Start** verwiesen wird. Die `Package.ini`-Vorlagendatei wird in einem ThinApp-Projekt zur Basis für die `Package.ini`-Datei.

Verwendung

```
snapshot.exe Snap1.snapshot –SuggestProject Snap2.snapshot OutputTemplate.ini
```

Beispiele

```
Snapshot Start.snapshot –SuggestProject End.snapshot Template.ini
```

ThinApp erfordert sämtliche Parameter.

Erstellen des ThinApp-Projekts aus der Template Package.ini-Datei

Das Dienstprogramm `snapshot.exe` erstellt die ThinApp-Projektdatei von der `Package.ini`-Vorlagendatei.

Verwendung

```
snapshot.exe Template.ini –GenerateProject OutDir [–Config ConfigFile.ini]
```

Beispiele

```
Snapshot Template.ini –GenerateProject C:\MyProject
              Snapshot Template.ini –GenerateProject C:\MyProject –Config MyExclusions.ini
```

`–Config ConfigFile.ini` ist optional. Die Konfigurationsdatei gibt Verzeichnisse oder Registrierungsunterschlüssel an, die vom Projekt ausgeschlossen werden sollen. Wenn Sie keine Konfigurationsdatei angeben, verwendet ThinApp die `snapshot.ini`-Datei.

Anzeigen von Inhalten einer Snapshot-Datei

Das Dienstprogramm `snapshot.exe` listet die Inhalte der Snapshot-Datei.

Verwendung

```
snapshot.exe SnapshotFileName.snapshot –Print
```

Beispiele

```
Snapshot Start.snapshot -Print
```

ThinApp erfordert sämtliche Parameter.

Beispiele für snapshot.exe-Befehle

Tabelle 7-1 beschreibt Beispielbefehle für das Dienstprogramm `snapshot.exe`. Die Parameter unterscheiden nicht zwischen Groß- und Kleinschreibung. Die Befehle sind aus Platzgründen in der Befehlsspalte mehrzeilig dargestellt.

Tabelle 7-1. snapshot.exe-Beispielbefehle

Befehl	Beschreibung
`snapshot C:\Capture.snapshot`	Kapselt einen vollständigen Snapshot der lokalen Laufwerke und der Registrierung in der Datei `C:\Capture.snapshot`.
`snapshot C:\Capture.snapshot C:\ E:\`	Kapselt einen vollständigen Snapshot der Laufwerke `C:\` und `E:\`. ThinApp kapselt keine Registrierungsdaten.
`snapshot C:\Capture.snapshot C:\data.snapshot C:\ HKEY_LOCAL_MACHINE`	Kapselt einen vollständigen Snapshot des Laufwerkes `C:\` und der gesamten `HKEY_CLASSES_ROOT`-Registrierungsunterstruktur.
`snapshot C:\Original.snapshot –Diff C:\NewEnvironment.snapshot C:\MyProject`	Generiert ein ThinApp-Projektverzeichnis mithilfe des Vergleichs zwischen zwei Snapshots.
`snapshot Original.snapshot –DiffPrint NewEnvironment.snapshot`	Zeigt die Unterschiede zwischen zwei gekapselten Snapshots an.
`snapshot C:\data.snapshot C:\ HKEY_LOCAL_MACHINE`	Speichert den Zustand des Computer-Dateisytems und der Registrierung.
`snapshot C:\start.snapshot –diffprint C:\end.snapshot`	Vergleicht zwei gespeicherte Zustände.
`snapshot C:\start.snapshot –print`	Druckt die Inhalte eines gespeicherten Zustandes.
`snapshot C:\start.snapshot –SuggestProject C:\end.snapshot C:\project.ini`	Generiert ein ThinApp-Projekt mithilfe des Vergleichs zwischen zwei gespeicherten Zuständen.

Erstellen eines Projekts ohne den Setup Capture-Assistenten

Sie können das Dienstprogramm `snapshot.exe` aus der Befehlszeile verwenden, statt des Setup Capture-Assistenten, der das Dienstprogramm `snapshot.exe` im Hintergrund ausführt. Das Befehlszeilen-Dienstprogramm ist nützlich, um eine große Anzahl an Anwendungen zu verpacken oder die ThinApp-Projekterstellung zu automatisieren. Der typische Speicherort für das Dienstprogramm `snapshot.exe` lautet `C:\Programme\VMware\VMware ThinApp\snapshot.exe`.

Der Snapshot-Prozess erstellt eine Kopie aller Registrierungseinträge auf dem System und in den Dateisystem-Metadaten. Zu den Dateisystem-Metadaten gehören Pfad, Dateiname, Attribut, Größe und Zeitstempel; tatsächliche Dateidaten werden jedoch ausgeschlossen.

So erstellen Sie ein Projekt mit dem Befehlszeilen-Dienstprogramm snapshot.exe

1 Speichern Sie einen Snapshot der aktuellen Computerkonfiguration auf der Festplatte.

```
snapshot.exe C:\Start.snapshot
```

2 Installieren Sie die Anwendung und nehmen Sie die erforderlichen manuellen Systemänderungen vor.

3 Speichern Sie einen Snapshot der neuen Computerkonfiguration auf der Festplatte.

```
snapshot.exe C:\End.snapshot
```

4 Generieren Sie eine `Package.ini`-Vorlagendatei.

```
snapshot.exe C:\Start.snapshot –SuggestProject C:\End.snapshot C:\Template.ini
```

ThinApp verwendet die Vorlagendatei, um die endgültige `Package.ini`-Datei zu generieren. Die Vorlagendatei enthält eine Liste aller erkannten ausführbaren Dateieinstiegspunkte und `Package.ini`-Parameter. Wenn Sie Ihr eigenes Script schreiben und den Setup Capture-Assistenten ersetzen, verwenden Sie die `Package.ini`-Vorlagendatei, um die Einstiegspunkte auszuwählen, und die `Package.ini`-Parameter wie `InventoryName` zu erhalten oder anzupassen.

5 Generieren eines ThinApp-Projekts.

```
snapshot.exe C:\Template.ini –GenerateProject C:\MyProjectDirectory
```

6 (Optional) Löschen Sie die temporären Dateien `C:\Start.snapshot`, `C:\End.snapshot` und `C:\Template.ini`.

7 (Optional) Um mehrere Projekte mit unterschiedlichen Konfigurationen zu generieren, verwenden Sie die ursprüngliche `Start.snapshot`-Datei erneut und wiederholen Sie den Vorgang ab Schritt 2.

Anpassen der Snapshot.ini-Datei

Die `snapshot.ini`-Konfigurationsdatei gibt an, welche Registrierungsschlüssel von einem ThinApp-Projekt auszuschließen sind, wenn eine Anwendung gekapselt wird.

Wenn Sie zum Beispiel Internet Explorer 7 verwenden, müssen Sie ThinApp möglicherweise veranlassen, folgende Registrierungsschlüssel zu kapseln:

- `HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections`
- `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings`

Wenn die `snapshot.ini`-Datei den `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections`-Schlüssel standardgemäß ausschließt, können Sie diesen Schlüssel aus der `snapshot.ini`-Datei entfernen, um zu gewährleisten, dass ThinApp den Schlüssel in die Kapselung einbezieht.

Wenn Sie die `snapshot.ini`-Datei nicht anpassen, lädt der Snapshot-Prozess die Datei von einem dieser Speicherorte:

- `Anwendungsdaten\Thinapp\snapshot.ini`

  Dieser Speicherort ist das `AppData`-Verzeichnis der Benutzer.

- `C:\Programme\VMware\VMWare Thinapp\snapshot.ini`

  Dies ist der Speicherort, von dem aus ThinApp das Dienstprogramm `snapshot.exe` ausführt.

ThinApp-Dateisystemformate und Makros 8

ThinApp speichert die Unterschiede zwischen Snapshots während des Setup Capture-Prozesses in einem virtuellen Dateisystem und einer virtuellen Registrierung. Das virtuelle Dateisystem verwendet Ordnermakros, um Speicherorte für Windows-Shell-Ordner darzustellen.

Diese Informationen über das virtuelle Dateisystem beinhalten folgende Themen:

- „Virtuelle Dateisystemformate“ auf Seite 107
- „ThinApp-Ordnermakros“ auf Seite 107

Virtuelle Dateisystemformate

ThinApp generiert die folgenden virtuellen Dateisystemformate:

- Build

  Durch den Setup Capture-Prozess wird dieses Format aus Dateien direkt vom physischen Dateisystem generiert. ThinApp verwendet Ordnermakros, um Speicherorte für Windows-Shell-Ordner darzustellen.

- Eingebettet

  Die Datei `build.bat` löst einen Build-Prozess aus, der ein schreibgeschütztes Dateisystem in ausführbare Dateien einbettet. Die ausführbaren Dateien stellen Clientcomputern blockbasiertes Streaming zur Verfügung. ThinApp komprimiert das Dateisystem.

- Sandbox

  Bei Ausführung der gekapselten Anwendung wird die Verzeichnisstruktur mit Lese- und Schreibrechten generiert, in der Dateiinformationen enthalten sind, die von der Anwendung modifiziert werden. Zu den Dateimodifikationen, die ThinApp zum Extrahieren eingebetteter virtueller Dateien in die Sandbox veranlassen, gehören folgende Vorgänge:

  - Änderung des Zeitstempels oder der Attribute einer Datei
  - Öffnen einer Datei mit Schreibzugriff
  - Abschneiden einer Datei
  - Umbenennen oder Verschieben einer Datei

Eingebettete und Sandbox-Dateisysteme verwenden Ordnermakros, damit Dateipfade dynamisch während der Laufzeit erweitert werden können.

ThinApp-Ordnermakros

ThinApp verwendet Makros, um Dateisystempfade darzustellen, die sich ändern können, wenn virtualisierte Anwendungen auf verschiedenen Windows-Betriebssystemen oder Computern laufen. Durch die Verwendung von Makros wird es möglich, die gemeinsam genutzten Profilinformationen einer Anwendung sofort in verschiedene Betriebssysteme zu migrieren.

Sie können beispielsweise eine Anwendung auf einem System kapseln, das `C:\WINNT` als Windows-Verzeichnis verwendet, und die Anwendung auf einem System mit `C:\Windows` als Windows-Verzeichnis bereitstellen. ThinApp wandelt `C:\WINNT` während der Kapselung dieses Systems in `%SystemRoot%` um und erweitert während der Laufzeit dieses Systems `%SystemRoot%` zu `C:\Windows`.

Registriert eine Anwendung DLLs auf `C:\winnt\system32`, während sie unter Windows 2000 ausgeführt wird, kann der Benutzer die Anwendung schließen und sich auf einem Computer mit dem Betriebssystem Windows XP anmelden. Auf dem Computer mit Windows XP sind die Dateien scheinbar unter `C:\Windows\system32` vorhanden und alle zugehörigen Registrierungsschlüssel verweisen auf `C:\windows\system32`.

Unter Windows Vista verschiebt ThinApp Windows SxS DLLs und Richtlinieninformationen und passt sie an Windows Vista an, anstatt Dateipfadstile von Windows XP zu verwenden. Diese Funktion ermöglicht es den meisten Anwendungen, auf aktualisierte oder ältere Betriebssysteme zu migrieren.

ThinApp bietet SxS-Unterstützung für Anwendungen die unter Windows 2000 ausgeführt werden, auch wenn das Basisbetriebssystem dies nicht tut. Diese Unterstützung ermöglicht den meisten in Windows XP gekapselten Anwendungen die unveränderte Ausführung unter Windows 2000.

Liste der ThinApp-Makros

ThinApp verwendet die Datei `shfolder.dll`, um den Speicherort von Shell-Ordnern zu ermitteln. Ältere Versionen der Datei `shfolder.dll` bieten keine Unterstützung für einige Makronamen.

Makros, für die `shfolder.dll`-Version 5.0 oder höher erforderlich ist, sind unter anderem `%ProgramFilesDir%`, `%Common AppData%`, `%Local AppData%`, `%My Pictures%` und `%Profile%`.

Makros, für die `shfolder.dll`-Version 6.0 oder höher erforderlich ist, sind unter anderem `%My Videos%`, `%Personal%` und `%Profiles%`.

Tabelle 8-1 listet die verfügbaren Ordnermakros.

Tabelle 8-1. Ordnermakros

Makroname	Typischer Speicherort
%AdminTools%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme\Verwaltung
%AppData%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Anwendungsdaten
%CDBurn Area%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Anwendungsdaten\Microsoft\CD Burning
%Common AdminTools%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme\Verwaltung
%Common AppData%	C:\Dokumente und Einstellungen\All Users\Anwendungsdaten
%Common Desktop%	C:\Dokumente und Einstellungen\All Users\Desktop
%Common Documents%	C:\Dokumente und Einstellungen\All Users\Dokumente
%Common Favorites%	C:\Dokumente und Einstellungen\All Users\Favoriten
%Common Programs%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme
%Common StartMenu%	C:\Dokumente und Einstellungen\All Users\Startmenü
%Common Startup%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme\Autostart
%Common Templates%	C:\Dokumente und Einstellungen\All Users\Vorlagen
%Cookies%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Cookies
%Desktop%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Desktop
%Drive_c%	C:\
%Drive_m%	M:\
%Favorites%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Favoriten
%Fonts%	C:\Windows\Fonts
%History%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Verlauf

Tabelle 8-1. Ordnermakros (Fortsetzung)

Makroname	Typischer Speicherort
%Internet Cache%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temporary Internet Files
%Local AppData%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Anwendungsdaten
%My Pictures%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien\Eigene Bilder
%My Videos%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien\Eigene Videos
%NetHood%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Netzwerkumgebung
%Personal%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien
%PrintHood%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Druckumgebung
%Profile%	C:\Dokumente und Einstellungen\<Benutzer_Name>
%Profiles%	C:\Dokumente und Einstellungen
%Program Files Common%	C:\Programme\Gemeinsame Dateien
%ProgramFilesDir%	C:\Programme
%Programs%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme
%Recent%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Zuletzt verwendete Dokumente
%Resources%	C:\Windows\Resources
%Resources Localized%	C:\Windows\Resources\<Sprache_ID>
%SendTo%	C:\Dokumente und Einstellungen\<Benutzer_Name>\SendTo
%Startup%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme\Autostart
%SystemRoot%	C:\Windows
%SystemSystem%	C:\Windows\System32
%TEMP%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temp
%Templates%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Vorlagen

Verarbeitung von %SystemRoot% in einer Terminaldienste-Umgebung

Eine Terminaldienste-Umgebung verfügt über ein freigegebenes Windows-Verzeichnis, wie beispielsweise `C:\Windows`, und ein privates Windows-Verzeichnis, wie beispielsweise `C:\Dokumente und Einstellungen\User\Windows`. In dieser Umgebung verwendet ThinApp das benutzerspezifische Verzeichnis für `%SystemRoot%`.

Erstellen von ThinApp-Skripts 9

Skripts ändern das Verhalten von virtuellen Anwendungen dynamisch. Sie können vor dem Starten einer mit ThinApp paketierten Anwendung oder nach dem Beenden der Anwendung einen benutzerdefinierten Code erstellen. Sie können Skripts zur Authentifizierung der Benutzer und zum Laden der Konfigurationsdateien von einer physischen in eine virtuelle Umgebung verwenden.

Rückruffunktionen führen während bestimmter Ereignisse Codes aus. Wenn die Anwendungen untergeordnete Prozesse erstellen, verwenden Sie Rückruffunktionen, um den Code nur im übergeordneten Hauptprozess auszuführen.

API-Funktionen führen ThinApp-Funktionen aus und interagieren mit der ThinApp-Laufzeit. API-Funktionen können Benutzer authentifizieren und den Start von Anwendungen für nicht autorisierte Benutzer verhindern.

Das Hinzufügen von Skripts zu Ihrer Anwendung beinhaltet das Erstellen einer ANSI-Textdatei mit der Dateierweiterung `.vbs` im Projektverzeichnis der Stammanwendung. Das Stammprojektverzeichnis ist das gleiche Verzeichnis, das die Datei `Package.ini` enthält. Während des Build-Prozess fügt ThinApp die Skriptdateien zu der ausführbaren Datei hinzu und führt jede der Skriptdateien während der Laufzeit aus.

ThinApp verwendet VBScript zum Ausführen von Skriptdateien. Informationen über VBScript finden Sie in der Dokumentation für Microsoft VBScript. Sie können VBScript zum Zugriff auf die COM-Steuerelemente, die auf dem Hostsystem oder in dem virtuellen Paket registriert sind, verwenden.

Dieser Abschnitt umfasst die folgenden Themen:

- „Rückruffunktionen“ auf Seite 111
- „Implementieren von Skripts in einer ThinApp-Umgebung“ auf Seite 112
- „API-Funktionen“ auf Seite 115

Rückruffunktionen

Rückruffunktionen können unter bestimmten Bedingungen ausgeführt werden. So können beispielsweise Rückruffunktionen nur dann einen Skriptcode ausführen, wenn eine Anwendung gestartet oder beendet wird.

Namen von Rückruffunktionen:

- `OnFirstSandboxOwner` – Wird nur aufgerufen, wenn eine Anwendung zum ersten Mal die Sandbox sperrt. Dieser Rückruf wird nicht aufgerufen, wenn eine zweite Kopie der gleichen Anwendung die gleiche Sandbox verwendet, während die erste Kopie ausgeführt wird. Wenn die erste Anwendung einen Unterprozess startet und beendet wird, sperrt der zweite Unterprozess die Sandbox und verhindert, dass dieser Rückruf ausgeführt wird, bis alle Unterprozesse beendet wurden und die Anwendung erneut ausgeführt wird.
- `OnFirstParentStart` – Wird aufgerufen, bevor eine ausführbare ThinApp-Datei ausgeführt wird, unabhängig davon, ob die Sandbox gleichzeitig von einer anderen gekapselten ausführbaren Datei verwendet wird.

- `OnFirstParentExit` – Wird aufgerufen, wenn der erste übergeordnete Prozess beendet ist. Wenn ein übergeordneter Prozess einen untergeordneten Prozess ausführt und beendet wird, wird dieser Rückruf auch dann aufgerufen, wenn der untergeordnete Prozess weiterhin ausgeführt wird.
- `OnLastProcessExit` – Wird aufgerufen, wenn der letzte Prozess, der von der Sandbox ausgeführt wird, beendet wird. Falls ein übergeordneter Prozess einen untergeordneten Prozess ausführt und beendet wird, wird dieser Rückruf aufgerufen, wenn der untergeordnete Prozess beendet wird.

Das folgende Beispiel für einen Rückruf zeigt die Funktionen für `OnFirstSandboxOwner` und `OnFirstParentExit`.

```
------------------------example.vbs ---------------------------------
Function OnFirstSandboxOwner
Msgbox "The sandbox owner is: " + GetCurrentProcessName
End Function

Function OnFirstParentExit
msgbox "Quiting application: " + GetCurrentProcessName
End Function

msgbox "This code will execute for all parent and child processes"
-----------------------------------------------------------------------
```

Implementieren von Skripts in einer ThinApp-Umgebung

Unter folgenden Umständen kann das Implementieren eines Skripts sinnvoll sein:

- Timeout einer Anwendung an einem bestimmten Datum.
- Ausführen einer `.bat`-Datei von einer Netzwerkfreigabe innerhalb der virtuellen Umgebung.
- Ändern der virtuellen Registrierung.
- Importieren der `.reg`-Datei während der Laufzeit.
- Anhalten eines virtuellen Dienstes, wenn die Hauptanwendung beendet wird.
- Kopieren einer externen Systemkonfigurationsdatei in die virtuelle Umgebung beim Start.

So implementieren Sie ein Skript

1 Speichern Sie den Skriptinhalt in eine Nur-Text-Datei mit der `.vbs`-Erweiterung und legen Sie sie im gleichen Verzeichnis wie Ihre `Package.ini`-Datei ab.

   Sie können einen beliebigen Dateinamen verwenden. ThinApp fügt bei der Build-Zeit des Pakets alle `.vbs`-Dateien zum Paket hinzu.

2 Erstellen Sie die Anwendung erneut.

Beispiel: .bat

Das folgende Skript führt eine externe `.bat`-Datei von einer Netzwerkfreigabe innerhalb der virtuellen Umgebung aus. Die `.bat`-Datei führt Änderungen an der virtuellen Umgebung durch, indem sie Dateien kopiert, Dateien löscht oder mithilfe von `regedit /s regfile.reg` Registrierungsänderungen durchführt. Führen Sie dieses Skript nur für den ersten übergeordneten Prozess aus. Wenn Sie dieses Skript für andere Prozesse ausführen, führt jede Kopie des Dienstprogramms `cmd.exe` das Skript aus und löst eine unendliche Rekursion aus.

```
Function OnFirstParentStart
 Set Shell = CreateObject("Wscript.Shell")
 Shell.Run "\\jcdesk2\test\test.bat"
End Function
```

Beispiel: Timeout

Das folgende Skript verhindert, dass eine Anwendung nach einem bestimmten Datum weiterhin ausgeführt wird. Das VBS-Datum verwendet unabhängig vom Gebietsschema das Format `#mm/dd/yyyy#`.

Diese Überprüfung wird beim Start des übergeordneten sowie beim Start aller untergeordneten Prozesse durchgeführt.

```
if Date >= #03/20/2007# then
 msgbox "This application has expired, please contact Administrator"
 ExitProcess 0
end if
```

Ändern der virtuellen Registrierung

Die folgende Skriptprozedur ändert die virtuelle Registrierung während der Laufzeit, um einen externen ODBC-Treiber von dem gleichen Verzeichnis zu laden, in dem sich die ausführbare Paketdatei befindet.

So ändern Sie die Registrierung

1 Suchen Sie den Pfad zu den ausführbaren Paketdateien.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

2 Suchen Sie den letzten Slash in dem Pfad und kopieren Sie die Zeichen vor dem Slash.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

3 Bilden Sie einen neuen Pfad zu der `ODBC DLL`-Datei, die sich außerhalb des Pakets befindet.

```
DriverPath=SourcePath + "tsodbc32.dll"
```

4 Ändern Sie die virtuelle Registrierung, so dass sie auf diesen Speicherort zeigt.

```
Set WSHShell = CreateObject("Wscript.Shell")
WSHShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\Transoft ODBC Driver\Driver,"
DriverPath
```

Diese Änderung führt dazu, dass die Anwendung die DLL von einem externen Speicherort aus lädt.

Beispiel: .reg

Mit dem folgenden Skript werden die Registrierungswerte von einer externen `.reg`-Datei während der Laufzeit in die virtuelle Registrierung importiert.

```
Function OnFirstParentStart
ExecuteVirtualProcess "regedit /s C:\tmp\somereg.reg"
End Function
```

Beispiel: Anhalten eines Dienstes

Mit dem folgenden Skript wird ein virtueller oder systemeigener Dienst angehalten, wenn die Hauptanwendung beendet wird.

```
Function OnFirstParentExit
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run "net stop ""iPod Service"""
End Function
```

Beispiel: Kopieren einer Datei

In den folgenden Skriptabschnitten wird gezeigt, wie eine Konfigurationsdatei, die sich im gleichen Verzeichnis befindet wie die gekapselte ausführbare Datei, bei jedem Anwendungsstart in das virtuelle Dateisystem kopiert wird. Dieses Skript ist nützlich für eine externe Konfigurationsdatei, die nach der Bereitstellung einfach zu bearbeiten ist. Da die Kopierfunktion jedes Mal ausgeführt wird, wenn Sie die Anwendung ausführen, werden alle Änderungen an der externen Version in der virtuellen Version reflektiert.

Wird beispielsweise Ihre gekapselte ausführbare Datei von `\\server\share\myapp.exe` ausgeführt, sucht dieses Skript nach einer Konfigurationsdatei, die sich am Speicherort `\\server\share\config.ini` befindet und kopiert sie in den Speicherort des virtuellen Dateisystems `C:\Programme\Meine Anwendung\config.ini`.

Indem Sie diesen Code in die Funktion `OnFirstParentStart` einfügen, wird sie nur einmal pro Ausführung des Skripts aufgerufen. Ansonsten wird sie für jeden untergeordneten Prozess ausgeführt.

```
Function OnFirstParentStart
```

ThinApp richtet `TS_ORIGIN` ein, um den vollständigen Pfad zu einem gekapselten ausführbaren Dateipaket anzugeben. Eine virtuelle Anwendung richtet die Variable `TS_ORIGIN` auf dem physischen Pfad des primären Datencontainers ein. Wenn Sie eine virtuelle Anwendung haben, die aus `main.exe` und `shortcut.exe` besteht, befinden sich beide Dateien in `C:\VirtApp`. Wenn Sie die Datei `main.exe` ausführen, wird `TS_ORIGIN var` auf `C:\VirtApp\main.exe` eingerichtet. Wenn Sie die Datei `shortcut.exe` ausführen, wird die Umgebungsvariable `TS_ORIGIN` auf `C:\VirtApp\main.exe` eingerichtet. Die Umgebungsvariable wird immer auf den primären Datencontainer eingerichtet, auch wenn Sie eine Verknüpfung erstellen. Wenn Sie VBScripts ausführen, die in dem Paket enthalten sind, ist die Variable bereits eingerichtet und für die Skripts verfügbar.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

Sie können den Dateinamen von `TS_ORIGIN` trennen, indem Sie nach dem letzten Backslash suchen und alle Zeichen nach diesem Backslash entfernen.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Die Quelldatei, die in die virtuelle Umgebung zu kopieren ist, ist der Pfad für das Paket plus `config.ini`.

```
SourceFile = SourcePath + "Config.ini"
```

Der Speicherort, in den kopiert werden soll, kann für jeden Computer anders sein, wenn das Verzeichnis `Programme` einem anderen Speicherort als `C:\` zugeordnet wurde. Mit dem folgenden Aufruf kann ThinApp ein Makro ausführen, um den korrekten Speicherort für den lokalen Computer aufzufinden.

```
DestFile = ExpandPath("%ProgramFilesDir%\MyApplication\Config.ini")
```

Verwenden Sie den Dateiparameter `systemObject`, um zu überprüfen, ob die Quelldatei vorhanden ist.

```
Set objFSO = CreateObject("Scripting.filesystemObject")
If objFSO.FileExists(SourceFile) Then
```

Wenn die Quelldatei vorhanden ist, kopieren Sie sie in das virtuelle Dateisystem. Das virtuelle Verzeichnis `%ProgramFilesDir%\MyApplication` befindet sich im Paket.

```
objFSO.CopyFile SourceFile, DestFile, TRUE
End if
End Function
```

Hinzufügen eines Wertes zur Systemregistrierung

Mit dieser Skriptprozedur wird ein Wert zu der physischen Systemregistrierung hinzugefügt.

So fügen Sie einen Wert zur Systemregistrierung hinzu

1 Erstellen Sie eine `.reg`-Datei und führen Sie den Befehl `regedit /s` als externen Prozess aus, der auf die Systemregistrierung anstelle der virtuellen Registrierung zugreift.

```
Function OnFirstParentStart
```

2 Erstellen Sie die `.reg`-Datei an einem Speicherort, bei dem der Parameter `IsolationMode` auf `Merged (Zusammengeführt)` eingerichtet ist, damit die virtuelle Umgebung mit diesem Skript auf sie zugreifen kann und die physische Umgebung mit dem Befehl `regedit /s`.

```
RegFileName = ExpandPath("%Personal%\thin.reg")
Set fso = CreateObject("Scripting.filesystemObject")
Set RegFile = fso.CreateTextFile(RegFileName, true)
```

Das Verzeichnis `%Personal%` ist ein Verzeichnis, das standardmäßig den Isolationsmodus „Merged (Zusammengeführt)“ besitzt.

3 Erstellen Sie die `.reg`-Datei.

```
RegFile.WriteLine("Windows Registry Editor Version 5.00")
RegFile.WriteBlankLines(1)
RegFile.WriteLine("[HKEY_CURRENT_USER\Software\Thinapp\demo]") RegFile.WriteLine(chr(34) and
"InventoryName" and chr(34) and "=" and chr(34) and GetBuildOption("InventoryName") and
chr(34))
RegFile.Close
```

4 Fügen Sie die Informationen zur Systemregistrierung hinzu.

```
RegEditPid = ExecuteExternalProcess("regedit /s " & chr(34) & RegFileName & chr(34))
WaitForProcess RegEditPid, 0
```

Warten Sie, bis der Prozess beendet ist.

5 Bereinigen Sie die Umgebung.

```
fso.DeleteFile(RegFileName)
End Function
```

API-Funktionen

Sie können API-Funktionen verwenden, mit denen ThinApp zum Fertigstellen von Vorgängen wie dem Laden von DLLs als virtuelle DLLs, zum Konvertieren von Pfaden aus dem Makroformat ins Systemformat und zum Ausführen von Befehlen innerhalb der virtuellen Umgebung angewiesen wird.

AddForcedVirtualLoadPath

Mit der Funktion `AddForcedVirtualLoadPath(Path)` wird ThinApp angewiesen, alle DLLs von dem festgelegten Pfad als virtuelle DLLs zu laden, auch wenn sie sich nicht im Paket befinden.

Verwenden Sie diese Funktion, wenn die Anwendung externe DLLs laden muss, die von DLLs innerhalb des Pakets abhängen.

Sie können die `ForcedVirtualLoadPaths`-Parameter in der `Package.ini`-Datei verwenden, um dasselbe Ergebnis wie diese API-Funktion zu erzielen. Siehe „ForcedVirtualLoadPaths“ auf Seite 68.

Parameter

`Path`

[in] Der Dateiname oder Pfad für DLLs, die als virtuelle DLLs geladen werden sollen.

Beispiele

Sie können alle DLLs als virtuelle DLLs laden, die sich im gleichen Verzeichnis wie die ausführbare Datei befinden.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

TS_ORIGIN ist der Pfad, von dem aus die ausführbare Datei ausgeführt wird.

Sie können den Dateinamen von TS_ORIGIN löschen, indem Sie nach dem letzten Backslash suchen und alle Zeichen nach diesem Backslash entfernen.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Sie können ThinApp anweisen, alle DLLs im gleichen oder einem niedrigeren Verzeichnis zu laden, in dem sich die ausführbare Quelldatei befindet.

```
AddForcedVirtualLoadPath(SourcePath)
```

Dieser Prozess ermöglicht Ihnen, zusätzliche Dateien in der SourcePath-Struktur abzulegen, damit sie Importvorgänge für virtuelle DLLs auflösen.

ExitProcess

Die Funktion `ExitProcessExitCode` beendet den aktuellen Prozess und setzt den spezifizierten Fehlercode.

Parameter

`ExitCode`

[in] Der zu setzende Fehlercode. Diese Informationen könnten für einen übergeordneten Prozess zur Verfügung stehen. Ein Wert 0 gibt an, dass kein Fehler vorliegt.

Beispiele

Sie können den Prozess beenden und den Erfolg anzeigen.

```
ExitProcess 0
```

Wenn der Prozess beendet ist, empfängt das Skriptsystem den Rückruf für die Funktion `OnLastProcessExist`. Alle geladenen DLLs führen einen Code für das Beenden aus, um die Umgebung zu bereinigen.

ExpandPath

Die Funktion `ExpandPath(InputPath)` konvertiert einen Pfad aus dem Makroformat ins Systemformat.

Parameter

`InputPath`

[in] Ein Pfad im Makroformat.

Ausgabe

Der erweiterte Makropfad im Systemformat.

Beispiele

Path = `ExpandPath("%ProgramFilesDir%\Myapp.exe")`

Path = `C:\Programme\myapp.exe`

Alle Makropfade müssen die Zeichen % und # durch #25 und #23 ersetzen.

Path = `ExpandPath("%ProgramFilesDir%\FilenameWithPercent#25.exe")`

Dies wird erweitert auf `C:\Programme\FileNameWithPercent%.exe`.

ExecuteExternalProcess

Die Funktion `ExecuteExternalProcess(CommandLine)` führt einen Befehl außerhalb der virtuellen Umgebung aus. Sie können diese Funktion verwenden, um Änderungen am physischen System vorzunehmen.

Parameter

`CommandLine`

[in] Repräsentation der Anwendung und der Befehlszeilenparameter zur Ausführung außerhalb der virtuellen Umgebung.

Ausgabe

Ganzzahlige Prozess-ID. Sie können die Prozess-ID mit der Funktion `WaitForProcess` verwenden. Siehe „WaitForProcess“ auf Seite 122.

Beispiele

```
ExecuteExternalProcess("C:\WINDOWS\system32\cmd.exe /c copy C:\systemfile.txt
C:\newsystemfile.txt")
```

Sie können einen Befehl ausführen, der Anführungszeichen in der Befehlszeile erfordert.

```
ExecuteExternalProcess("regsvr32 /s " & chr(34) & "C:\Program Files\my.ocx" & chr(34))
```

ExecuteVirtualProcess

Die Funktion `ExecuteVirtualProcess(CommandLine)` führt einen Befehl innerhalb der virtuellen Umgebung aus. Sie können diese Funktion verwenden, um Änderungen an der virtuellen Umgebung vorzunehmen.

Parameter

`CommandLine`

[in] Repräsentation der Anwendung und der Befehlszeilenparameter zur Ausführung außerhalb der virtuellen Umgebung.

Ausgabe

Ganzzahlige Prozess-ID. Sie können die Prozess-ID mit der Funktion `WaitForProcess` verwenden. Siehe „WaitForProcess“ auf Seite 122.

Beispiele

```
ExecuteVirtualProcess("C:\WINDOWS\system32\cmd.exe /c copy C:\systemfile.txt C:\virtualfile.txt")
```

Sie können einen Befehl ausführen, der Anführungszeichen in der Befehlszeile erfordert.

```
ExecuteVirtualProcess("regsvr32 /s " & chr(34) & "C:\Program Files\my.ocx" & chr(34))
```

GetBuildOption

Die Funktion `GetBuildOption(OptionName)` gibt den Wert einer Einstellung aus, die im Abschnitt `[BuildOptions]` der Datei `Package.ini`, die für gekapselte Anwendungen verwendet wird, festgelegt wird.

Parameter

`OptionName`

[in] Name der Einstellung.

Ausgabe

Diese Funktion gibt einen Zeichenfolgenwert aus. Wenn der angeforderte Optionsname nicht existiert, gibt die Funktion einen leeren Zeichenfolgenwert aus ("").

Beispiele

```
Package.ini enthält:
[BuildOptions]
CapturedUsingVersion=4.0.1–2866
```

In einer VBS-Datei wird die folgende Zeile angezeigt:

```
Value = GetBuildOption("CapturedUsingVersion")
```

GetFileVersionValue

Die Funktion `GetFileVersionValue(Filename, Value)` gibt den Wert für die Versionsinformationen von Dateien wie bestimmten DLL, OCX oder ausführbaren Dateien aus. Sie können diese Funktion dazu verwenden, die interne Versionsnummer einer DLL zu bestimmen oder DLL-Informationen über den Copyright-Inhaber oder einen Produktnamen abzurufen.

Parameter

`Filename`

[in] Der Name des Dateinamens, dessen Versionsinformationen abgerufen werden.

Value

[in] Der Name des Wertes, der von dem Versionsinformationsabschnitt der bestimmten Datei abgerufen werden soll.

Sie können von den meisten DLLs die folgenden Werte abrufen:

- Comments
- InternalName
- ProductName
- CompanyName
- LegalCopyright
- ProductVersion
- FileDescription
- LegalTrademarks
- PrivateBuild
- FileVersion
- OriginalFilename
- SpecialBuild

Ausgabe

Diese Funktion gibt einen Zeichenfolgenwert aus. Wenn der angeforderte Dateiname nicht vorhanden ist oder die Funktion den bestimmten Wert nicht in der Datei finden kann, gibt die Funktion eine leere Zeichenfolge aus ("").

Beispiele

```
FileVersion = GetFileVersionValue("C:\windows\system32\kernel32.dll," "FileVersion")

if FileVersion = "1.0.0.0" then
                MsgBox "This is Version 1.0!"

End if
```

GetCommandLine

Mit der Funktion `GetCommandLine` wird auf die Befehlszeilenparameter zugegriffen, die an das laufende Programm übergeben werden.

Ausgabe

Diese Funktion gibt eine Zeichenfolge aus, die die Befehlszeilenargumente darstellt, die an das derzeit laufende Programm übergeben werden, einschließlich der ursprünglichen ausführbaren Datei.

Beispiele

```
MsgBox "The command line for this EXE was " + GetCommandLine
```

GetCurrentProcessName

Mit der Funktion `GetCurrentProcessName` wird auf den vollständigen virtuellen Pfadnamen des derzeitigen Prozesses zugegriffen.

Ausgabe

Diese Funktion gibt eine Zeichenfolge aus, die den vollständigen ausführbaren Pfadnamen innerhalb der virtuellen Umgebung darstellt. Unter den meisten Umständen lautet dieser Pfad `C:\Programme\...`, auch wenn die Paketquelle von einer Netzwerkfreigabe ausgeführt wird.

Beispiele

```
MsgBox "Running EXE path is " + GetCurrentProcessName
```

GetOSVersion

Die Funktion `GetOSVersion()` gibt Informationen über die aktuelle Windows-Version aus.

Parameter

Diese Funktion hat keine Parameter.

Ausgabe

Diese Funktion gibt eine Zeichenfolge in folgendem Format aus: `MAJOR.MINOR.BUILD_NUMBER.PLATFORM_ID OS_STRING`.

`MAJOR` kann einen der folgenden Werte annehmen:

Windows Vista	6
Windows Server 2008	6
Windows Server 2003	5
Windows XP	5
Windows 2000	5
Windows NT 4.0	4

MINOR kann einen der folgenden Werte annehmen:

Windows Vista	0
Windows Server 2008	0
Windows Server 2003	2
Windows XP	1
Windows 2000	0
Windows NT 4.0	0
Windows NT 3,51	51

BUILD_NUMBER ist die Versionsnummer des Betriebssystems.

PLATFORM_ID weist einen der folgenden Werte zu:

- Value = 1 für Windows Me, Windows 98 oder Windows 95 (Windows 95-basiertes Betriebssystem)
- Value = 2 für Windows Server 2003, Windows XP, Windows 2000 oder Windows NT. (Windows NT-basiertes Betriebssystem)

OS_STRING stellt die Informationen über das Betriebssystem dar, wie beispielsweise Service Pack 2.

Beispiele

```
if GetOSVersion() = "5.1.0.2 Service Pack 2"
               then MsgBox "You are running on Windows XP Service Pack 2!"
endif
```

GetEnvironmentVariable

Die Funktion GetEnvironmentVariable(Name) gibt die Umgebungsvariable aus, die der Variablen Name zugewiesen wurde.

Parameter

Name

[in] Der Name der Umgebungsvariable, für die der Wert abgerufen wird.

Ausgabe

Diese Funktion gibt den Zeichenfolgenwert aus, der der Umgebungsvariable Name zugewiesen wurde.

Beispiele

```
MsgBbox "The package source EXE is " + GetEnvironmentVariable("TS_ORIGIN")
```

RemoveSandboxOnExit

Mit der Funktion RemoveSandboxOnExit(YesNo) kann durch eine Ja/Nein-Auswahl bestimmt werden, ob die Sandbox gelöscht wird, wenn der letzte untergeordnete Prozess beendet wird.

Wenn Sie den Parameter RemoveSandboxOnExit in der Datei Package.ini auf 1 setzen, lautet das Standard-Bereinigungsverhalten für das Paket Yes (Ja). Sie können das Bereinigungsverhalten auf No (Nein) einrichten, indem Sie RemoveSandboxOnExit mit dem Wert 0 aufrufen. Wenn Sie den Eintrag RemoveSandboxOnExit=1 in der Datei Package.ini nicht ändern, lautet das Standard-Bereinigungsverhalten für das Paket No (Nein). Sie können das Bereinigungsverhalten zu Yes (Ja) ändern, indem Sie RemoveSandboxOnExit mit dem Wert 1 aufrufen.

Parameter

`Yes No`

[in] Wollen Sie nach dem Ende des letzten Prozesses eine Bereinigung durchführen? 1=Yes (Ja), 0=No (Nein)

Beispiele

Das folgende Beispiel aktiviert die Bereinigung.

```
RemoveSandboxOnExit 1
```

Das folgende Beispiel deaktiviert die Bereinigung.

```
RemoveSandboxOnExit 0
```

SetEnvironmentVariable

Die Funktion `SetEnvironmentVariable(Name, Value)` richtet den Wert einer Umgebungsvariable ein.

Parameter

`Name`

[in] Der Name der Umgebungsvariable zum Speichern des Wertes.

`Value`

[in] Der zu speichernde Wert.

Beispiele

```
SetEnvironmentVariable "PATH", "C:\Windows\system32"
```

SetfileSystemIsolation

Die Funktion `Setfile systemIsolation(Directory, IsolationMode)` richtet den Isolationsmodus eines Verzeichnisses ein.

Parameter

`Directory`

[in] Vollständiger Pfad des Verzeichnisses, dessen Isolationsmodus eingerichtet werden soll.

`IsolationMode`

[in] Einzurichtender Isolationsmodus.

1 = WriteCopy
2 = Merged
3 = Full

Beispiele

Sie können den Isolationsmodus „Merged (Zusammengeführt)“ für das temporäre Verzeichnis einrichten.

```
Setfile systemIsolation GetEnvironmentVariable("TEMP"), 2
```

SetRegistryIsolation

Die Funktion `SetRegistryIsolation(RegistryKey, IsolationMode)` richtet den Isolationsmodus eines Registrierungsschlüssels ein.

Parameter

`RegistryKey`

[in] Der Registrierungsschlüssel, auf den der Isolationsmodus eingerichtet werden soll. Beginnen Sie mit `HKLM` für `HKEY_LOCAL_MACHINE`, `HKCU` für `HKEY_CURRENT_USER` und `HKCR` für `HKEY_CLASSES_ROOT`.

`IsolationMode`

[in] Einzurichtender Isolationsmodus.

1 = WriteCopy
2 = Merged
3 = Full

Beispiele

Sie können den Isolationsmodus „Full (Voll)“ für `HKEY_CURRENT_USER\Software\Thinapp\Test` einrichten.

```
SetRegistryIsolation "HKCU\Software\Thinapp\Test," 3
```

WaitForProcess

Die Funktion `WaitForProcess(ProcessID, TimeOutInMilliSeconds)` wartet, bis die Prozess-ID die Ausführung beendet hat.

Parameter

`ProcessID`

[in] Die zu beendende Prozess-ID. Die Prozess-ID kommt entweder von `ExecuteExternalProcess` oder von `ExecuteVirtualProcess`.

`TimeOutInMilliSeconds`

[in] Die maximale Zeitspanne, in der auf das Beenden des Prozesses gewartet wird, bis der Vorgang fortgesetzt wird. Ein Wert 0 legt INFINITE (unendlich) fest.

Ausgabe

Diese Funktion gibt eine Ganzzahl aus.

0 = Timeout fails (Timeout fehlgeschlagen)
1 = Process exits (Prozess ist beendet)
2 = Process does not exist or security is denied (Prozess ist nicht vorhanden oder Sicherheit ist verweigert)

Beispiele

```
id = ExecuteExternalProcess("C:WINDOWS\system32\cmd.exe")
WaitForProcess(id, 0)
```

ThinApp – Überwachung und Problembehandlung 10

Verwenden Sie Protokoll-Monitor zum Generieren von Trace-Dateien und zur Problembehandlung innerhalb der ThinApp-Umgebung. Protokoll-Monitor ist ausschließlich mit einer gekapselten Anwendung, die dieselbe Version von ThinApp verwendet, kompatibel.

Dieser Abschnitt umfasst die folgenden Themen:

- „Bereitstellen von Informationen für den technischen Support“ auf Seite 123
- „Protokoll-Monitor-Vorgänge“ auf Seite 124
- „Problembehandlung bei bestimmten Anwendungen“ auf Seite 131

Bereitstellen von Informationen für den technischen Support

Zur Problembehandlung innerhalb einer ThinApp-Umgebung benötigt der technische Support von VMware folgende Angaben von Ihnen:

- Eine schrittweise Schilderung der Vorgänge, die Sie vornahmen, als das Problem auftrat.
- Informationen über die Hostkonfiguration. Geben Sie das Windows-Betriebssystem an, die Verwendung von Terminal Server oder Citrix Xenapp und alle erforderlichen Programme, die auf dem nativen System installiert sind.
- Kopien der Trace-Dateien von Protokoll-Monitor. Siehe „Protokoll-Monitor-Vorgänge“ auf Seite 124.
- Eine genaue Kopie des Kapselungsordners und seines gesamten Inhalts. Beziehen Sie die kompilierten ausführbaren Dateien aus dem `/bin`-Unterordner nicht mit ein.
- Eine Beschreibung des erwarteten und des tatsächlichen Verhaltens der Anwendung.
- (Optional) Kopien der Anwendungen, die Sie gekapselt haben. Schließen Sie die Server-Komponentenkonfiguration für Oracle Server oder Active Directory mit ein.
- (Optional) Native oder physische Dateien oder Einstellungen für Registrierungsschlüssel, die für das Problem bedeutsam sein könnten.
- (Optional) Systemdienste oder erforderliche Gerätetreiber.
- (Optional) Die virtuelle Maschine, die den Fehler nachbildet. Der VMware-Support benötigt sie möglicherweise, wenn die Support-Kontaktperson nicht in der Lage ist, das Problem nachzustellen.
- (Optional) Eine oder mehrere WebEx-Sitzungen, um das Debugging in Ihrer Systemumgebung zu vereinfachen.

Protokoll-Monitor-Vorgänge

Protokoll-Monitor erfasst detailliert und chronologisch die Aktivitäten ausführbarer, von der gekapselten Anwendung gestarteter Dateien. Protokoll-Monitor fängt Protokollnamen, Adressen, Parameter und Rückgabewerte für jeden Funktionsaufruf der ausführbaren Zieldateien oder DLLs ab und protokolliert sie. Protokoll-Monitor erfasst folgende Aktivitäten:

- Win32 API-Aufrufe von Anwendungen, die im virtuellen Betriebssystem von ThinApp ausgeführt werden.
- Potenzielle Fehler, Ausnahmen und Sicherheitsereignisse innerhalb der Anwendung.
- Alle DLLs, die von der Anwendung und den Adressbereichen geladen werden.

Die generierten Protokolldateien können größer als 100 MB sein, je nachdem, wie lange die Anwendung mit Protokoll-Monitor ausgeführt wird und wie ausgelastet eine Anwendung ist. Der einzige Grund, warum Protokoll-Monitor für eine Anwendung ausgeführt wird, ist die Erfassung der Trace-Dateien. Trace-Dateien sind wesentlich für die Problembehandlung, da innerhalb der Trace-Datei mehrere Einträge analysiert und korreliert werden.

Problembehandlung mit Protokoll-Monitor

Sie können Protokoll-Monitor für die grundlegende Problembehandlung einsetzen.

Problembehandlung mithilfe der ThinApp-Protokolle

1 Beenden Sie die gekapselte Anwendung vor der Untersuchung.

2 Wählen Sie auf dem Computer, auf dem Sie die Anwendung gekapselt haben, **Start** > **Alle Programme** > **VMware** > **ThinApp Log Monitor**.

Um Protokoll-Monitor von einem Arbeitscomputer aus zu starten, kopieren Sie die Dateien `log_monitor.exe`, `logging.dll` und `Setup Capture.exe` von `C:\Programme\VMware\VMware ThinApp` auf den Arbeitscomputer und doppelklicken Sie auf die Datei `log_monitor.exe`.

3 Starten Sie die gekapselte Anwendung.

Mit dem Start der Anwendung wird in der Protokoll-Monitor-Liste ein neuer Eintrag angezeigt. Protokoll-Monitor zeigt für jede neue Trace-Datei einen neuen Eintrag an. Eine einzelne Datei muss nicht unbedingt an einen einzelnen Vorgang gekoppelt sein.

4 Beenden Sie die Anwendung, sobald ein Fehler auftritt.

5 Generieren Sie Protokolle für jede Trace-Datei, die Sie untersuchen möchten.

a Wählen Sie die `.trace`-Datei aus der Liste aus.

b Klicken Sie auf **Trace-Textbericht generieren (Generate text trace report)**.

Vom übergeordneten Prozess generierte untergeordnete Prozesse sind im gleichen Protokoll gespeichert. Mehrere, unabhängige Prozesse sind nicht im selben Protokoll gespeichert.

ThinApp generiert eine `.trace`-Datei. Protokoll-Monitor konvertiert die binäre `.trace`-Datei in eine `.txt`-Datei.

6 (Optional) Öffnen Sie die `.txt`-Datei in einem Text-Editor und überprüfen Sie die Informationen. Unter bestimmten Umständen ist die `.txt`-Datei zu groß, um mit dem Text-Editor geöffnet zu werden.

7 Zippen Sie die `.txt`-Dateien und senden Sie die Dateien an VMware-Support.

Ausführen von erweiterten Protokoll-Monitor-Vorgängen

Zu den erweiterten Protokoll-Monitor-Vorgängen gehört das Beenden von Anwendungen oder das Löschen von Trace-Dateien. Ist eine Anwendung ausgelastet oder weist sie bei einer bestimmten Aktion nur eine verlangsamte Leistung auf, können Sie die Vorgänge `Anhalten (Suspend)` und `Fortsetzen (Resume)` ausführen, um die Protokolle eines bestimmten Zeitraums zu erfassen. Die resultierende Protokolldatei ist kleiner als die typische Protokolldatei und einfacher zu analysieren. Auch bei der Verwendung der Vorgänge `Anhalten (Suspend)` und `Fortsetzen (Resume)` kann die ursprüngliche Ursache für einen Fehler außerhalb des Zeitfensters liegen. Die Vorgänge `Anhalten (Suspend)` und `Fortsetzen (Resume)` sind global und wirken sich auf sämtliche Anwendungen aus.

Um weitere Informationen über die Verwendung dieser Optionen zu erhalten, wenden Sie sich bitte an den VMware-Support.

So führen Sie erweiterte Protokoll-Monitor-Vorgänge aus

1 Beenden Sie die gekapselte Anwendung vor der Untersuchung.

2 Wählen Sie auf dem Computer, auf dem Sie die Anwendung gekapselt haben, **Start** > **Programme** > **VMware** > **ThinApp Log Monitor**.

   Um Protokoll-Monitor von einem Arbeitscomputer aus zu starten, kopieren Sie die Dateien `log_monitor.exe`, `logging.dll` und `Setup Capture.exe` von `C:\Programme\VMware\VMware ThinApp` auf den Arbeitscomputer und doppelklicken Sie auf die Datei `log_monitor.exe`.

3 (Optional) Erfassen Sie die Protokolle eines bestimmten Zeitraums, um einen zeitlich genau festgelegten Fehler zu suchen.

   a Aktivieren Sie das Kontrollkästchen **Anhalten (Suspend)**.

   b Starten Sie die gekapselte Anwendung und führen Sie sie bis zum Punkt aus, an dem der Fehler auftritt oder die Leistungsminderung beginnt.

   c Deaktivieren Sie im Protokoll-Monitor das Kontrollkästchen **Anhalten (Suspend)**, um den Protokollierungsvorgang wiederaufzunehmen.

      Sie können das Verhalten der Anwendung prüfen, um das Problem zu isolieren.

   d Aktivieren Sie das Kontrollkästchen **Anhalten (Suspend)**, um den Protokollierungsvorgang anzuhalten.

4 (Optional) Wählen Sie in der Trace-Dateiliste die zu löschende Datei und klicken Sie auf **Datei löschen (Delete File)**.

5 (Optional) Klicken Sie auf **Abbrechen (Kill App)**, um einen laufenden Prozess anzuhalten.

6 (Optional) Aktivieren Sie das Kontrollkästchen **Komprimieren (Compress)**, um die Größe einer Trace-Datei zu verringern.

   Dieser Vorgang vermindert die Leistung der Anwendung.

7 (Optional) Generieren Sie einen Trace-Dateibericht.

   a Wählen Sie in der Dateiliste eine Trace-Datei aus, geben Sie einen Trace-Dateinamen ein oder klicken Sie auf **Durchsuchen (Browse)**, um eine Trace-Datei auf Ihrem System auszuwählen.

   b (Optional) Geben Sie den Namen des Ausgabeberichts ein oder verändern Sie den Namen des Ausgabeberichts.

   c Klicken Sie auf **Trace-Textbericht generieren (Generate text trace report)**, um einen Bericht zu erstellen.

      Sie können die Datei mit einem Text-Editor anzeigen, der Zeilenumbrüche nach dem UNIX-System unterstützt.

Ermitteln von Fehlern

Die ThinApp-Protokollierung stellt eine große Menge an Informationen bereit. Die folgenden Tipps sollen fortgeschrittene Anwender bei der Fehleruntersuchung unterstützen:

- Lesen Sie den Abschnitt `Potential Errors Detected (Erkannte potenzielle Fehler)` der `.txt`-Trace-Datei.

  Die Einträge zeigen möglicherweise keine Fehler an. ThinApp listet jeden Win32 API-Aufruf auf, bei dem sich der Windows-Fehlercode verändert hat.

- Prüfen Sie die von den Anwendungen generierten Ausnahmen.

  Ausnahmen können auf Fehler hinweisen. Zu den Ausnahmetypen gehören C++ und .NET. Die Trace-Datei zeichnet den Ausnahmetyp auf und die DLL, die die Ausnahme generiert. Wenn eine Anwendung, wie .NET oder Java, eine Ausnahme aus einem selbstgenerierenden Code erzeugt, zeigt die Trace-Datei ein unbekanntes Modul an.

  Das folgende Beispiel ist ein `.trace`-Eintrag für eine Ausnahme.

  ```
  *** Exception EXCEPTION_ACCESS_VIOLATION on read of 0x10 from unknown_module:0x7c9105f8
  ```

  Wenn Sie auf eine Ausnahme stoßen, überprüfen Sie den Anfangsteil der Trace-Datei auf die Ursache der Ausnahme. Ignorieren Sie Gleitkommaausnahmen, die Virtual Basic 6-Anwendungen bei normaler Verwendung generieren.

- Betrachen Sie die untergeordneten Prozesse.

  Protokoll-Monitor erstellt für jeden Prozess eine `.trace`-Datei. Startet eine Anwendung mehrere untergeordnete Prozesse, müssen Sie ermitteln, welcher Prozess den Fehler verursacht. Unter bestimmten Umständen, beispielsweise bei prozessunabhängigem COM, verwendet eine übergeordnete Anwendung COM, um einen untergeordneten Prozess zu starten, führt eine Funktion remote aus und fährt mit der Ausführung von Funktionen fort.

- Wenn Sie Anwendungen von einer Netzwerkfreigabe aus starten, die zwei Prozesse generiert, ignorieren Sie den ersten Prozess.

  ThinApp begegnet der herabgesetzten Leistung von Symantec Antivirus-Anwendungen durch erneutes Starten der Prozesse.

- Suchen Sie die in den Dialogfeldern angezeigte Fehlermeldung.

  Einige Anwendungen rufen die Funktion MessageBox Win32 API auf, um unerwartete Fehler während der Laufzeit anzuzeigen. Sie können in einer Trace-Datei nach einem `MessageBox` suchen oder nach dem Inhalt der Zeichenfolge, die in der Fehlermeldung angezeigt wird, und feststellen, welche Anwendung zuletzt ausgeführt wurde, bevor das Dialogfeld angezeigt wurde.

- Engen Sie die Auswahl an Aufrufen von einer bestimmten DLL und einem bestimmten Thread ein.

  Das Protokollformat legt die DLL und den Thread fest, die einen Aufruf auslösen. Die Aufrufe von System-DLLs können häufig ignoriert werden.

Protokollformat

Eine Trace-Datei enthält folgende Abschnitte:

- Systemkonfiguration

  Dieser Abschnitt enthält Informationen über das Betriebssystem, die Laufwerke, die installierten Programme, die Umgebungsvariablen, die Prozessliste, die Dienste und Treiber.

  Die Informationen beginnen mit der Zeichenfolge `Dump started on (Abbildsicherung gestartet um)` und enden mit der Zeichenfolge `Dump ended on (Abbildsicherung beendet um)`.

- Kopfzeile

  In diesem Abschnitt werden Kontextinformationen angezeigt, zum Beispiel über den durch Protokoll-Monitor nachverfolgten Prozess. Einige der angezeigten Attribute zeigen die Anmeldeoptionen, Adressbereiche beim Laden der Laufzeit des Betriebssystems und die Makrozuordnung zu bestehenden Systempfaden.

  ThinApp markiert den Beginn des Kopfzeilenabschnitts mit der Sequenzziffer 000001. Unter normalen Umständen markiert ThinApp das Ende dieses Abschnitts mit einer Meldung über das Dienstprogramm Application Sync.

- Textkörper

  Dieser Abschnitt enthält die Trace-Aktivität beim Starten und Ausführen von Vorgängen durch die Anwendung. Jede Zeile stellt einen Funktionsaufruf dar, den die ausführbaren Zieldateien oder eine der DLLs vornehmen.

  Der Abschnitt beginnt mit dem Eintrag `New Modules detected in memory (Im Arbeitsspeicher erkannte neue Module)`, gefolgt von der Modulliste `SYSTEM_LOADED`. Der Abschnitt endet mit dem Eintrag `Modules Loaded (Geladene Module)`.

- Zusammenfassung

  Dieser Abschnitt enthält Module, die von der gekapselten Anwendung geladen werden, potenzielle Fehler und ein Profil der 150 langsamsten Aufrufe.

  Der Abschnitt beginnt mit der Meldung `Modules Loaded (Geladene Module)`.

Allgemeines Format der API-Protokollmeldungen

Die folgende Meldung ist ein Beispiel für das Format von API-Aufrufen.

```
000257 0a88 mydll.dll :4ad0576d–>kernel32.dll:7c81b1f0 SetConsoleMode (IN HANDLE
hConsoleHandle=7h, IN DWORD dwMode=3h)
000258 0a88 mydll.dll :4ad0576d<–kernel32.dll:7c81b1f0 SetConsoleMode –>BOOL=1h ()
```

Dieses Beispiel enthält folgende Einträge:

- `000257` steht für die Protokolleintragsnummer. Jeder Protokolleintrag besitzt eine eigene Nummer.
- `0a88` steht für die Kennung des aktuell ausgeführten Threads. Hat die Anwendung nur einen Thread, bleibt diese Nummer unverändert. Zeichnen zwei oder mehrere Threads Daten in der Protokolldatei auf, kann man die Thread-Kennung verwenden, um die zu einem Thread gehörenden aufeinanderfolgenden Aktionen nachzuverfolgen, da ThinApp Protokolleinträge in der Reihenfolge ihres Auftretens verzeichnet.
- `mydll.dll` steht für die DLL, die den API-Aufruf vornimmt.
- `4ad0576d` steht für die Rücksprungadresse für den API-Aufruf, den `mydll.dll` vornimmt. Unter normalen Umständen ist die Rücksprungadresse die Adresse im Code, bei der der Aufruf durchgeführt wurde.
- –> zeigt den Beginn des Funktionsaufrufs an. Für den Protokolleintrag der begonnenen Ausführung des Funktionsaufrufs zeigt ThinApp die Eingabeparameter an. Diese Parameter sind in- und in-/out-Parameter.
- <– zeigt die Rückkehr vom Funktionsaufruf zur ursprünglichen Einsprungstelle an. Für Protokolleinträge zur Funktionsrückkehr zeigt ThinApp die Ausgabeparameter an. Diese Parameter sind out- und in/out-Parameter.
- `kernel32.dll` steht für die DLL, von der der API-Aufruf ausgeführt wird.
- `7c81b1f0` steht für die Adresse der API-Funktion innerhalb der kernel32.dll, von der der Aufruf ausgeführt wird. Indem man `kernel32.dll` an der `7c81b1f0`-Adresse disassembliert, kann man den Code für die Funktion SetConsoleMode anzeigen.
- –>`BOOL=1h` zeigt an, dass die API-Funktion den Wert 1 zurückgibt und der Rückgabewert vom Typ BOOL ist.

Informationen zum Anwendungsstart

Die folgenden Einträge bieten grundlegende Informationen über die Anwendung, beispielsweise den Modulnamen und die Prozess-ID (PID), sowie über Protokoll-Monitor, beispielsweise die Version und die Optionen.

```
000001 0a88 Logging started for Module=C:\test\cmd_test\bin\cmd.exe
Using archive=
PID=0xec
CommandLine = cmd
000002 0a88 Logging options: CAP_LEVEL=9 MAX_CAP_ARY=25 MAX_CAP_STR=150
MAX_NEST=100
VERSION=3.090

000003 0a88 System Current Directory = C:\test\cmd_test\bin Virtual Current Directory =
C:\test\cmd_test\bin

000004 0a88 |start_env_var| =::=::\
000005 0a88 |start_env_var| =C:=C:\test\cmd_test\bin
000006 0a88 |start_env_var| =ExitCode=00000000
000007 0a88 |start_env_var| ALLUSERSPROFILE=C:\Dokumente und Einstellungen\All Users.WINDOWS
...
...
...
```

Liste der während der Laufzeit in den Arbeitsspeicher geladenen DLLs

Der Abschnitt `Modules loaded (Geladene Module)` findet sich nahe dem Ende der Protokolldatei und beschreibt die DLLs, die während der Laufzeit in den Arbeitsspeicher geladen werden, sowie die DLL-Adressen. Die Informationen zeigen an, ob Windows oder ThinApp die DLLs lädt.

Dieses Beispiel enthält eine Zusammenfassung der Länge der längsten Aufrufe sowie die folgenden Einträge:

- `SYSTEM_LOADED` zeigt an, dass Windows die DLL lädt. Die Datei muss auf der Festplatte vorhanden sein.
- `MEMORY_MAPPED_ANON` zeigt an, dass ThinApp die DLL lädt. ThinApp kann die Datei vom virtuellen Dateisystem laden.
- `46800000-46873fff` zeigt den Adressbereich im virtuellen Arbeitsspeicher an, in dem sich die DLL befindet.
- `PRELOADED_BY_SYSTEM` und `PRELOADED_MAP` sind Duplikateinträge, die sich auf den Arbeitsspeicheradressbereich beziehen, in dem die ausführbare Image-Datei im Arbeitsspeicher abgebildet wird.

```
---Modules loaded --
PRELOADED_MAP 00400000-00452fff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.exe
PRELOADED_BY_SYSTEM 00400000-00452fff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.exe
SYSTEM_LOADED 00400000-00452fff, C:\Test\AcroRd32.exe
MEMORY_MAPPED_ANON 013b0000-020affff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.dll

----Timing Report: list of slowest 150 objects profiled ---

8255572220 total cycles (2955.56 ms): |sprof| thinapp_LoadLibrary2

765380728 cycles (274.01 ms) on log entry 21753
428701805 cycles (153.48 ms) on log entry 191955
410404281 cycles (146.93 ms) on log entry 193969
.
.
... 438 total calls
7847975891 total cycles (2809.64 ms): |sprof| ts_load_internal_module
764794646 cycles (273.80 ms) on log entry 21753
426837866 cycles (152.81 ms) on log entry 191955
408570540 cycles (146.27 ms) on log entry 193969
```

```
.
.
... 94 total calls
4451728477 total cycles (1593.76 ms): |sprof| ts_lookup_imports
544327945 cycles (194.87 ms) on log entry 21758
385149968 cycles (137.89 ms) on log entry 193970
187246661 cycles (67.04 ms) on log entry 190210
.
.
... 34 total calls
1099873523 total cycles (393.76 ms): |sprof| new_thread_start
561664565 cycles (201.08 ms) on log entry 151922
531551734 cycles (190.30 ms) on log entry 152733
1619002 cycles (0.58 ms) on log entry 72875
```

Potenzielle Fehler

Der Abschnitt `Erkannte potenzielle Fehler (Potential Errors Detected)` zeigt die Protokolleinträge an, deren Zeichenfolgen Probleme mit drei Sternen (***) aufweisen. Informationen zur Interpretation dieses Abschnitts erhalten Sie unter „Ermitteln von Fehlern“ auf Seite 126.

```
----Potential Errors Detected ---

006425 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
006427 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL' flags=2
-> 0 (failed ***)
006428 0000089c nview.dll :1005b94b<-kernel32.dll:7c80ae4b *** LoadLibraryW -
>HMODULE=7c800000h () *** GetLastError() returns 2 [0]: The system cannot find the file
specified.
007062 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
010649 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL'
flags=2 -> 0 (failed ***)
019127 0000075c MSVCR80.dll :781348cc<-msvcrt.dll :77c10396 *** GetEnvironmentVariableA -
>DWORD=0h (OUT LPSTR lpBuffer=*0h <bad ptr>) *** GetLastError() returns 203 [0]: The system
could not find the environment option that was entered.
019133 0000075c MSVCR80.dll :78133003<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [203]: The specified procedure could not be found.
019435 0000075c MSVCR80.dll :78136e08<-dbghelp.dll :59a60360 *** Getfile type ->DWORD=0h ()
*** GetLastError() returns 6 [0]: The handle is invalid.
019500 0000075c MSVCR80.dll :78134481<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [0]: The specified procedure could not be found.
019530 0000075c MSVCR80.dll :78131dcd<-dbghelp.dll :59a603a1 *** GetModuleHandleA -
>HMODULE=0h () *** GetLastError() returns 126 [0]: The specified module could not be found.
```

Beispiel für die Problembehandlung beim Dienstprogramm cmd.exe

Im Problembehebungsbeispiel verpackt ThinApp das Dienstprogramm `cmd.exe` bei aktivierter Protokollierung. Dieses Beispiel zeigt, wie Sie durch Ausführen eines ungültigen Befehls das fehlerhafte Verhalten einer Anwendung simulieren können. Wenn das Dienstprogramm `cmd.exe` dazu aufgefordert wird, den Befehl `foobar` auszuführen, generiert das Dienstprogramm die Meldung `foobar is not recognized as an internal or external command (foobar wird nicht als interner oder externer Befehl erkannt)`. Sie können die Trace-Datei und den Abschnitt `Erkannte potenzielle Fehler (Potential Errors Detected)` überprüfen, um die API-Funktionen zu suchen, die den Code `GetLastError` veränderten.

Das Beispiel zeigt die Pfade `C:\test\cmd_test\bin\foobar.*`, `C:\WINDOWS\system32\foobar.*` und `C:\WINDOWS\foobar` als Speicherorte, an denen das Dienstprogramm `cmd.exe` nach dem Befehl `foobar` sucht.

Das Beispiel zeigt die Pfade `%drive_C%\test\cmd_test\bin`, `%SystemSystem%\foobar` und `%SystemRoot%\foobar` als Speicherorte im virtuellen Dateisystem, die ThinApp durchsucht.

```
----Potential Errors Detected ---
*** Unable to determine if any services need to be auto-started, error 2 (Es konnte nicht
ermittelt werden, ob Dienste automatisch gestartet werden müssen, Fehler 2)
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW ->HANDLE=ffffffffh .. *** GetLastError() returns 2 [203]: The system cannot
find the file specified.
*** FindFirstFileW 'C:\test\cmd_test\bin\foobar' -> INVALID_HANDLE_VALUE *** failed [FS
missing in view 0][fs entry not found %drive_C%\test\cmd_test\bin\foobar][fs entry not found
%drive_C%\test\cmd_test\bin]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system
probe C:\WINDOWS\system32\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing
in view 0][fs entry not found %SystemSystem%\foobar]
*** FindFirstFileW 'C:\WINDOWS\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system probe
C:\WINDOWS\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing in view
0][fs entry not found %SystemRoot%\foobar]
```

Ausführen von erweiterten Untersuchungen von cmd.exe-Protokolleinträgen

Eine gründlichere Untersuchung eines Eintrags aus dem Abschnitt `Potential Errors Detected (Erkannte potenzielle Fehler)` in einer Trace-Datei kann die Suche in der gesamten Trace-Datei von Protokoll-Monitor nach dem betreffenden Eintrag umfassen sowie die Prüfung der Systemaufrufe und der Bedingungen, die zum potenziellen Fehler führten.

Der folgende Eintrag für das Dienstprogramm `cmd.exe` im Abschnitt `Potenzielle Fehler (Potential Errors)` könnte beispielsweise eine gründlichere Untersuchung durch die Trace-Datei von Protokoll-Monitor erfordern.

```
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe
```

So führen Sie eine erweiterte Untersuchung des cmd.exe-Eintrags aus

1 Um zu ermitteln, warum das Dienstprogramm `cmd.exe c:\test\cmd_test\bin` sondiert, überprüfen Sie das Protokoll für diese Protokolleintragsnummer und stellen Sie fest, was sich vor dem Aufruf ereignete.

2 Um die Speicherorte zu ermitteln, wo das Dienstprogramm `cmd.exe` den Pfad `c:\test\cmd_test path` abruft, überprüfen Sie das Protokoll auf die Einträge `GetCurrentDirectoryW` und `GetFullPathNameW`.

```
000861 0a88 cmd.exe  :4ad01580->USERENV.dll :769c0396 GetCurrentDirectoryW (IN DWORD
nBufferLength=104h)
000862 0a88                     GetCurrentDirectoryW -> 0x14 (C:\test\cmd_test\bin)
000863 0a88 cmd.exe  :4ad01580<-USERENV.dll :769c0396 GetCurrentDirectoryW ->DWORD=14h
(OUT LPWSTR lpBuffer=*4AD34400h->L"C:\test\cmd_test\bin")
000864 0a88 cmd.exe  :4ad05b74->ole32.dll :774e03f0 Getfile type (IN HANDLE hFile=7h)
000865 0a88                     Getfile type 7 -> 0x2
000866 0a88 cmd.exe  :4ad05b74<-ole32.dll :774e03f0 Getfile type ->DWORD=2h ()
.
.
001533 0a88 cmd.exe  :4ad01b0d<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001534 0a88 cmd.exe  :4ad01b13->kernel32.dll:7c80ac0f SetErrorMode (IN UINT uMode=1h)
001535 0a88 cmd.exe   :4ad01b13<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001536 0a88 cmd.exe  :4ad01b24->IMM32.DLL :7639039b GetFullPathNameW (IN LPCWSTR
lpFileName=*1638C0h->L"," IN DWORD nBufferLength=208h)
001537 0a88                     GetFullPathNameW . -> 20 (buf=C:\test\cmd_test\bin,
file_part=bin)
001538 0a88 cmd.exe  :4ad01b24<-IMM32.DLL :7639039b GetFullPathNameW ->DWORD=14h
(OUT LPWSTR lpBuffer=*163D60h->L"C:\test\cmd_test\bin," OUT *lpFilePart=*13D8D4h-
>*163D82h->L"bin")
.
.
```

```
001549 0a88 cmd.exe   :4ad01b5f->USERENV.dll :769c03fa FindFirstFileW (IN LPCWSTR
lpFileName=*1638C0h->L"C:\test\cmd_test\bin\foobar.*")
001550 0a88                    FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no
virtual or system matches]
```

Das Dienstprogramm `cmd.exe` erhält den ersten Speicherort durch den Aufruf `GetCurrentDirectoryW` und den zweiten Speicherort durch den Aufruf `GetFullPathNameW`, wobei "." den Pfad festlegt. Diese Aufrufe liefern den Pfad für das aktuelle Arbeitsverzeichnis zurück. Die Protokolldatei zeigt, dass das Dienstprogramm `cmd.exe` die Aufforderung `C:\test\cmd_test\bin>` erstellt. Das Dienstprogramm fragt die `PROMPT`-Umgebungsvariable ab, die `$P$G` zurückliefert, und verwendet die API-Funktion `WriteConsoleW`, um die Eingabeaufforderung auf dem Bildschirm anzuzeigen, nachdem `$P$G` intern zu`C:\test\cmd_test\bin>`erweitert wurde.

Problembehandlung bei bestimmten Anwendungen

Tipps für die Problembehandlung stehen für die Kapselung von Microsoft Outlook, Explorer.exe und Java Runtime Environment zur Verfügung.

Fehlerbehebung beim Setup der Registrierung für Microsoft Outlook

Microsoft Outlook speichert Kontoeinstellungen in Registrierungsschlüsseln und -dateien. Wenn Sie Microsoft Outlook zum ersten Mal starten, prüft das Programm, ob die Schlüssel vorhanden sind. Kann Microsoft Outlook die Schlüssel nicht lokalisieren, werden Sie aufgefordert, ein neues Konto zu erstellen.

Dieser Vorgang funktioniert in der virtuellen Umgebung fehlerfrei, wenn Microsoft Outlook nicht auf dem physischen System installiert ist. Hat der Anwender Microsoft Outlook bereits auf dem physischen System installiert, findet die gekapselte Version die Registrierungsschlüssel in der Systemregistrierung und verwendet diese Einstellungen. Sie müssen den Isolationsmodus „Full (Voll)“ für die Registrierungsschlüssel und -dateien verwenden, in denen Outlook seine Einstellungen speichert.

So legen Sie den Isolationsmodus „Full (Voll)“ für die Registrierungsschlüssel von Microsoft Outlook fest

1 Fügen Sie zur Datei `HKEY_CURRENT_USER.txt` die folgenden Einträge hinzu:

```
isolation_full HKEY_CURRENT_USER\Identities
isolation_full HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Windows Messaging Subsystem\Profiles
```

2 Erstellen Sie eine `##Attributes.ini`-Datei mit den folgenden Einträgen:

```
[Isolation]
DirectoryIsolationMode=Full
```

3 Platzieren Sie die `##Attributes.ini`-Datei in jedem der folgenden Unterverzeichnisse.

```
%AppData%\Microsoft\AddIns
%AppData%\Microsoft\Office
%AppData%\Microsoft\Outlook
%Local AppData%\Microsoft\FORMS
%Local AppData%\Microsoft\Outlook
```

4 (Optional) Erstellen Sie die Unterverzeichnisse, wenn sie nicht vorhanden sind.

Anzeigen von Anhängen in Microsoft Outlook

Sobald Sie einen Anhang zum Betrachten öffnen, erstellt Microsoft Outlook ein Standardverzeichnis zum Speichern von Anhängen. Der übliche Speicherort lautet: `C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temp\Temporary Internet Files\OLK<xxxx>`. Die letzten `xxxx` werden durch einen nach dem Zufallsprinzip erstellten Eintrag ersetzt.

Sie können Anhänge betrachten, wenn die Anwendung zum Betrachten in derselben virtuellen Sandbox wie Microsoft Outlook ausgeführt wird. Externe Anwendungen können möglicherweise die Datei zur Anzeige nicht finden, da Microsoft Outlook die Datei in der Sandbox speichert. Sie müssen den Isolationsmodus „Merged (Zusammengeführt)“ verwenden für das Verzeichnis, das die Anhänge speichert.

So legen Sie den Isolationsmodus „Merged (Zusammengeführt)“ für die Anzeige von Anhängen in Microsoft Outlook fest

1 Fügen Sie zur Datei `HKEY_CURRENT_USER.txt` einen Wert hinzu, der den Namen des Anhangverzeichnisses festlegt:

```
isolation_full
HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Outlook\Security
Value=OutlookSecureTempFolder
REG_SZ~%Profile%\Lokale Einstellungen\OutlookTempxxxx#2300
```

In diesem Beispiel steht 11.0 im Schlüsselnamen für Microsoft Outlook 2003.

2 Ersetzen Sie zur Erhöhung der Sicherheit die letzten vier `xxxx`-Zeichen durch zufällige alphanumerische Zeichen.

3 Erstellen Sie ein Verzeichnis, das im Registrierungsschlüssel `OutlookSecureTempFolder` Ihres ThinApp-Projekts benannt wird.

Erstellen Sie zum Beispiel das Verzeichnis `%Profile%\Lokale Einstellungen\OutlookTempxxxx`.

4 Erstellen Sie im Verzeichnis `%Profile%\Lokale Einstellungen\OutlookTempxxxx` eine `##Attributes.ini`-Datei mit den folgenden Einträgen:

```
[Isolation]
DirectoryIsolationMode=Merged
```

Starten von Explorer.exe in der virtuellen Umgebung

Durch das Ausführen einer Instanz des Dienstprogramms `explorer.exe` auf einem Windows-Betriebssystem wird das Hinzufügen eines Einstiegspunkts zu Windows Explorer und das Starten des Programms innerhalb der virtuellen Umgebung erschwert.

Mithilfe der folgenden Methoden können Sie innerhalb der virtuellen Umgebung ein Fenster von Windows Explorer öffnen:

- Fügen Sie zu iExplorer einen Einstiegspunkt hinzu und starten Sie die Datei mit dem `–E`-Parameter.

  Fügen Sie beispielsweise der Datei `Package.ini` die folgenden Einträge hinzu:

  ```
  [iexplore.exe]
  Shortcut=xxxx.exe
  Source=%ProgramFilesDir%\Internet Explorer\iexplore.exe
  CommandLine=%ProgramFilesDir%\Internet Explorer\iexplore.exe –E
  ```

- Fügen Sie den folgenden virtuellen Registrierungsschlüssel hinzu:

  ```
  isolation_full HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer
  Value=DesktopProcess
  REG_DWORD=#01#00#00#00
  ```

- Fügen Sie zur Datei `Package.ini` die folgenden Einträge hinzu:

  ```
  [explorer.exe]
  Shortcut=xxxxxx.exe
  Source=%SystemROOT%\explorer.exe
  ```

  Verwenden Sie diese Methode, um das virtuelle Dateisystem mit einer bekannten Oberfläche zu durchsuchen, und aktivieren Sie Dateitypzuordnungen ohne Systemänderungen, insbesondere bei der Verwendung von portablen Anwendungen. Sie können ohne Systemänderungen auf die Shell-integrierten Komponenten zugreifen.

Problembehandlung bei Versionskonflikten von Java Runtime Environment

Ein Konflikt kann auftreten, wenn eine Java-Version auf dem physischen System installiert ist und eine andere Version in einer gekapselten ausführbaren Datei enthalten ist. Aktualisierte Versionen von Java installieren eine Plug-In-DLL, die Internet Explorer lädt. Diese Plug-In-DLL überschreibt virtuelle Registrierungsschlüssel und führt zu Konflikten mit virtuellen Kopien älterer Java-Laufzeiten.

So verhindern Sie das Laden von Plug-In-DLLs durch Internet Explorer

Fügen Sie am Anfang der Datei `HKEY_LOCAL_MACHINE.txt` folgenden Eintrag hinzu:

```
isolation_full HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects
```

Glossar

A

Application Link
Ein Dienstprogramm, das voneinander abhängige Anwendungen während der Laufzeit mit einer Basisanwendung verknüpft und diese gemeinsam startet, sobald die Basisanwendung gestartet wird. Sie können das Dienstprogramm verwenden, um Komponentenpakete einzeln bereitzustellen und zu aktualisieren, anstatt alle Komponenten im selben Paket zu kapseln.

Application Sync
Ein Dienstprogramm zur Aktualisierung einer Anwendung, das neue verpackte Versionen auf einem Server oder in einem freigegebenem Netzwerk erkennt. Update-Einstellungen, wie die Überprüfung eines Update-Servers in bestimmten Zeitintervallen, können konfiguriert werden. ThinApp erkennt die neueste ausführbare Datei der Anwendung und lädt die Unterschiede herunter.

attributes.ini
Diese Datei wendet Konfigurationseinstellungen auf der Verzeichnisebene des Pakets statt auf der Ebene des gesamten Pakets an. Die `##Attributes.ini`-Einstellungen überschreiben alle `Package.ini`-Einstellungen.

B

Bestandsname
Ein Name, den ThinApp zur internen Nachverfolgung der Anwendung verwendet. Der Bestandsname legt den standardmäßigen Projektverzeichnisnamen fest und wird im Dialogfeld **Software** von Windows angezeigt.

Build
So konvertieren Sie ein ThinApp-Projekt in ein Paket. Ein Paket kann mithilfe des Setup Capture-Assistenten oder mit dem Dienstprogramm `build.bat` erstellt werden.

E

Einstiegspunkt
Eine ausführbare Datei zum Starten der gekapselten Anwendung. Eine Anwendung kann über mehrere Einstiegspunkte verfügen. Beispielsweise kann die Datei `Firefox.exe` als Einstiegspunkt für eine Mozilla Firefox-Anwendung verwendet werden. Die primäre Datencontainerdatei kann in einem Einstiegspunkt oder als `.dat`-Datei vorhanden sein.

I

Isolationsmodus
Eine Paketeinstellung, die den Lese- und Schreibzugriff auf die physische Umgebung festlegt. ThinApp verfügt über die Isolationsmodi: WriteCopy, Merged (Zusammengeführt) und Full (Voll).

K

kapseln
So paketieren Sie eine Anwendung in eine virtuelle Umgebung und legen Sie die anfänglichen Anwendungsparameter fest. ThinApp stellt den Setup Capture-Assistenten oder das Dienstprogramm `snapshot.exe` zur Erstellung eines portablen Anwendungspakets bereit, das unabhängig von dem Betriebssystem ist, auf dem es ausgeführt wird.

L

logging.dll
Ein Dienstprogramm, das `.trace`-Dateien generiert.

M

MSI
Ein Windows Installer-Container, der für Anwendungs-Bereitstellungstools nützlich ist. Sie können die gekapselte Anwendung als MSI-Datei anstatt einer ausführbaren Datei verteilen.

N

Nachüberprüfung
Zum Erstellen einer Abbildung oder eines Snapshots einer Maschine nach der Installation der Anwendung, die Sie kapseln möchten. Der Kapselungsvorgang speichert die Unterschiede zwischen den Vor- und Nachüberprüfungsabbildungen in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* Vorüberprüfung, Snapshot.

nativ
Bezieht sich eher auf die physische Umgebung als auf die virtuelle Umgebung. *Siehe auch* physisch.

Netzwerk-Streaming
Bei diesem Vorgang wird ein Paket von einem Zentralserver aus ausgeführt. ThinApp lädt nach Bedarf Blöcke der Anwendung herunter, um eine schnelle Verarbeitung und Anzeige zu gewährleisten.

neu aufgesetzte Maschine
Der Computer oder die virtuelle Maschine, nur mit dem elementaren Windows-Betriebssystem installiert, auf dem die Anwendung erfasst wird. Die Betriebssystemversion von Windows muss die früheste Version sein, mit der eine Anwendung ausgeführt werden soll.

P

package.ini
Die Datei, die die Konfigurationseinstellungen auf das Paket anwendet und im gekapselten Anwendungsordner gespeichert ist. Der Setup Capture-Assistent richtet die ursprünglichen Konfigurationseinstellungen ein.

Paket
Die virtuellen Anwendungsdateien, die vom ThinApp-Erstellungsvorgang generiert werden. Das Paket enthält die primäre Datencontainerdatei und Einstiegspunkt-Dateien zum Zugriff auf die Anwendung.

physisch
Bezieht sich auf den Computerarbeitsspeicher und das Dateisystem, in dem alle Standardprozesse von Windows ausgeführt werden. Abhängig von den Einstellungen des Isolationsmodus in ThinApp, können Vorgänge in der virtuellen Umgebung auf die physische Umgebung zugreifen. *Siehe auch* nativ, virtuell.

Primärer Datencontainer
Die Hauptdatei der virtuellen Anwendung. Die Datei ist eine `.exe`- oder eine `.dat`-Datei, in der die ThinApp-Laufzeit sowie das schreibgeschützte virtuelle Dateisystem und die virtuelle Registrierung enthalten sind. Der primäre Datencontainer muss im selben `/bin`-Verzeichnis zusammen wie alle untergeordneten ausführbaren Dateien der Anwendung gespeichert werden, da Einstiegspunkte auf die Informationen im primären Datencontainer zugreifen.

Projekt
Die Daten, die beim Kapselungsvorgang erzeugt werden, bevor ein Paket erstellt wird. Der Kapselungsprozess verwendet den Bestandsnamen als Standardprojektverzeichnisnamen. Sie können die Parameter in den Projektdateien anpassen, bevor Sie ein Anwendungspaket erzeugen. Eine gekapselte Anwendung kann erst bereitgestellt werden, nachdem ein Paket aus dem Projekt erstellt wurde.

Protokoll-Monitor
Ein Dienstprogramm, das die Aktivitäten ausführbarer Dateien, die von der gekapselten Anwendung gestartet werden, chronologisch erfasst. Die Datei `log_monitor.exe` ist ausschließlich mit gekapselten Anwendungen, die dieselbe Version von ThinApp verwenden, kompatibel.

S

Sandbox
Der physische Systemordner, der während der Laufzeit an der virtuellen Anwendung vorgenommene Benutzeränderungen speichert. Sobald die Anwendung gestartet wird, integriert ThinApp die Änderungen aus der Sandbox. Wenn Sie die Sandbox löschen, führt ThinApp die Anwendung zurück in den gekapselten Zustand. Der Standardspeicherort der Sandbox ist `%APPDATA%\Thinstall\<application_name>`.

sbmerge.exe
Ein Dienstprogramm, das inkrementelle Updates der Anwendungen erstellt, beispielsweise die Integrierung eines Plug-Ins oder einer Änderung auf einer Browser-Startseite. Das Dienstprogramm `sbmerge.exe` führt die in der Sandbox aufgezeichneten Laufzeitänderungen zurück in ein ThinApp-Projekt.

Snapshot
Eine Statuserfassung des Dateisystems und der Registrierung von Windows während der Kapselung der Anwendung. Der Setup Capture-Vorgang verwendet das Dienstprogramm `snapshot.exe`, um mit einem Snapshot den Systemzustand vor und nach der Installation der Anwendung zu erfassen und speichert die Unterschiede in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* Nachüberprüfung, Vorüberprüfung.

snapshot.exe
Ein Dienstprogramm, das Snapshots des Dateisystems und der Registrierung eines Computers erstellt und die Vor- und Nachüberprüfungsvorgänge während des Kapselungsvorgangs vereinfacht. Nur fortgeschrittene Benutzer, die die ThinApp-Funktionalität in andere Plattformen integrieren, werden dieses Dienstprogramm möglicherweise direkt verwenden. *Siehe auch* Nachüberprüfung, Vorüberprüfung, Snapshot.

snapshot.ini
Eine Konfigurationsdatei, die die Verzeichnisse und Unterschlüssel festlegt, die bei der Kapselung einer Anwendung von einem ThinApp-Projekt ausgeschlossen werden sollen. Diese Datei kann für Anwendungen angepasst werden.

T

template.msi
Eine Vorlage für MSI-Dateien, die Sie anpassen können, um die Bereitstellungsverfahren und -normen des Unternehmens einzuhalten. Zum Beispiel können Registrierungseinstellungen hinzufügt werden, die ThinApp als Bestandteil der Installation zu einem Clientcomputer hinzufügen soll.

thinreg.exe
Ein Dienstprogramm, das Dateitypzuordnungen erstellt, das **Start**-Menü und Verknüpfungen einrichtet und das Öffnen von Dateien vereinfacht. Sie müssen das Dienstprogramm `thinreg.exe` ausführen, um ausführbare Dateien zu registrieren. MSI-Dateien automatisieren den `thinreg.exe`-Registrierungsprozess.

tlink.exe
Ein Dienstprogramm, dass während des Kapselungsvorgangs Schlüsselmodule verknüpft.

V

vftool.exe
Ein Dienstprogramm, das das virtuelle Dateisystem während des Kapselungsvorgangs kompiliert.

virtuell
Bezieht sich auf die logische Datei und den Arbeitsspeicher, in dem eine gekapselte Anwendung ausgeführt wird. Prozesse in einer physischen Umgebung können nicht auf die virtuelle Umgebung zugreifen. *Siehe auch* physisch.

virtuelle Anwendung
Eine Anwendung, die gekapselt wird, damit sie portabel und unabhängig vom Betriebssystem ist, auf dem sie ausgeführt wird.

virtuelle Registrierung
Die Registrierung, wie sie von der gekapselten Anwendung gesehen wird.

virtuelles Dateisystem
Das Dateisystem, wie es von der gekapselten Anwendung gesehen wird.

Vorüberprüfung
Zum Erstellen einer Baselineabbildung oder eines Snapshots einer Maschine vor der Installation der Anwendung, die Sie kapseln möchten. Der Kapselungsvorgang speichert die Unterschiede zwischen den Vor- und Nachüberprüfungsabbildungen in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* Nachüberprüfung, Snapshot.

vregtool.exe
Ein Dienstprogramm, das die virtuelle Registrierung während des Kapselungsvorgangs kompiliert.

Index

Symbole

##Attributes.ini
- bearbeiten **25**
- Vergleichen mit Package.ini **25**, **58**

A

Active Directory
- Gruppenzugriff autorisieren **18**
- Package.ini-Parameter verwenden **36**
- Zugriff auf Anwendungen steuern **35**

Aktualisieren von Anwendungen, Methoden und Überlegungen **43**–**56**

Anwendungen
- aktualisieren **43**
- kapseln **15**
- Nutzungsstatistik **21**
- Streaminganforderungen und -empfehlungen **38**
- Überlegungen zur Sandbox in Bezug auf Upgrade-Vorgänge **55**
- Unterschied zwischen Application Sync und Application Link **43**
- von ThinApp nicht unterstützt **12**
- Zugriff steuern für Active Directory-Gruppen **35**

API-Parameter
- AddForcedVirtualLoadPath **115**
- ExecuteExternalProcess **116**
- ExecuteVirtualProcess **117**
- ExitProcess **116**
- ExpandPath **116**
- GetBuildOption **117**
- GetCommandLine **119**
- GetCurrentProcessName **119**
- GetEnvironmentVariable **120**
- GetFileVersionValue **118**
- GetOSVersion **119**
- RemoveSandboxOnExit **120**
- SetEnvironmentVariable **121**
- SetfileSystemIsolation **121**
- SetRegistryIsolation **121**
- WaitForProcess **122**

Application Link
- Ansicht von **48**
- Auswirkung auf Isolationsmodi **50**
- Datei- und Registrierungskollisionen **51**
- definieren **43**, **47**
- Definition für den Zugriff mit dem PermittedGroups-Parameter **50**
- Einrichten der verschachtelten Links **49**
- erforderliche Links **87**
- optionale Links **88**
- Parameter **86**
- Pfadnamenformate **86**
- Speichern mehrerer Versionen von verknüpften Anwendungen **51**
- Verknüpfen von Paketen mit Basisanwendungen unter Verwendung von Application Sync **51**
- Workflow-Beispiel **48**

Application Sync
- Aktualisieren der thinreg.exe-Registrierungen **45**
- Aktualisieren von Basisanwendungen mit verknüpften Paketen **51**
- Auswirkung auf ausführbare Dateien mit Einstiegspunkt **45**
- Auswirkung auf thinreg.exe **31**
- Bearbeiten von Parametern **44**
- Beibehalten des primären Datencontainernamens **45**
- definieren **43**
- Erzwingen der Updates mit appsync.exe-Befehlen **52**
- Kollidieren mit automatischen Update-Funktionen **43**
- Korrigieren der Updates **45**
- Parameter **88**

Ausschneide- und Einfügevorgänge, ThinApp-Beschränkungen **39**

B

bereitstellen
- Anwendungen auf Netzwerkfreigabe **30**
- Anwendungen mit Bereitstellungstools **29**
- ausführbare Dateien **30**
- MSI-Dateien **29**

Bereitstellungstools, unter Verwendung von MSI-Dateien **29**

Bestandsname, Zweck **21**

Betriebssysteme
- Unterstützung für **11**
- Verwenden der frühesten Version für die ThinApp-Installation **13**

C

cmd.exe, definieren **17**

Computer
- Definition von sauberem System **12**

virtuelle Maschine als sauberes System verwenden **13**

D

Datencontainer, *Siehe* primärer Datencontainer
DCOM-Dienste, Zugriff auf gekapselte Anwendungen **12**
Dienste
in Paketen starten und anhalten **36**
DLLs
Aufzeichnung durch Protokoll-Monitor **124**
in den Arbeitsspeicher laden **128**

E

Einstiegspunkte
Aktualisieren mit Application Sync **45**
definieren **17**
im Setup Capture-Assistent **17**
zur Problembehandlung **17**

G

Gerätetreiber, nicht kompatibel mit ThinApp **12**
globale Hook-DLLs, beschränkte Funktion mit ThinApp **12**

I

iexplore.exe, definieren **17**
Installation von ThinApp **13**
Isolationsmodi
ändern **61**
Beispielkonfiguration **41**
Full **61**
Merged (Zusammengeführt) **20**
Verwenden von Application Link **50**
WriteCopy **20**
Isolationsmodus „Merged“ **20**
Isolationsmodus „WriteCopy“ **20**

K

Kapselung von Anwendungen
Anforderungen und Abhängigkeiten **15**
Anforderungen zum Gebietsschema **15**
Bewertung von Anwendungsabhängigkeiten **15**
mit dem Dienstprogramm snapshot.exe **105**
mit dem Setup Capture-Assistenten **16–24**
Phasen der **15**
Komprimierung
ausführbarer Dateien **22**
für Trace-Dateien **125**

M

Microsoft Office
Anforderungen für die Kapselung **25**
kapseln **26**
Nachüberprüfungsoptionen **27**
Microsoft Vista, Bereitstellung von MSI-Dateien **35**
MSI-Dateien
Automatisieren des Dienstprogramms thinreg.exe **22**
Bereitstellung auf Microsoft Vista **35**
Datenbank aufbauen **33**
generieren **22**
Installationsverzeichnis überschreiben **35**
Package.ini ändern **34**
Parameter **91**
Parameter anpassen **33**

N

Netzwerk, Pakete per Stream übertragen **37**

P

Package.ini
AccessDeniedMsg **65**
Active Directory-Parameter **36**
AddPageExecutePermission **65**
allgemeine Parameter **24**
AllowExternalKernelModeServices **75**
AllowExternalProcessModifications **75**
AllowUnsuppportedExternalChildProcesses **76**
AnsiCodePage **82**
AppSyncClearSandboxOnUpdate **89**
AppSyncExpireMessage **89**
AppSyncExpirePeriod **89**
AppSyncUpdateFrequency **90**
AppSyncUpdateMessage **90**
AppSyncURL **90**
AppSyncWarningFrequency **90**
AppSyncWarningMessage **91**
AppSyncWarningPeriod **91**
AutoShutdownServices **76**
AutoStartServices **76**
Bearbeiten von Application Sync-Parametern **44**
BlockSize **77**
CachePath **72**
CapturedUsingVersion **81**
ChildProcessEnvironmentDefault **77**
ChildProcessEnvironmentExceptions **77**
CommandLine **83**
CompressionType **78**
DirectoryIsolationMode **60**
Disabled **83**
DisableTracing **80**
ExcludePattern **63**
ExternalCOMObjects **67**
ExternalDLLs **68**
FileTypes **62**
ForcedVirtualLoadPaths **68**
Icon **64**
InventoryName **96**
IsolatedMemoryObjects **69**
IsolatedSynchronizationObjects **69**

Konfigurieren der Application Link-Parameter **86**
Konfigurieren der Build-Parameter **63**
Konfigurieren der Laufzeitparameter **58**
Konfigurieren der Sicherheitsparameter **65**
Konfigurieren der Speicherparameter **72**
Konfigurieren von Application Sync-Parametern **88**
Konfigurieren von Datei- und Protokollzuordnungsparametern **62**
Konfigurieren von einzelnen Anwendungsparametern **83**
Konfigurieren von Größenparametern **77**
Konfigurieren von Isolationsparametern **60**
Konfigurieren von lokalen Parametern **82**
Konfigurieren von MSI-Parametern **91**
Konfigurieren von Objekt- und DLL-Parametern **67**
Konfigurieren von Protokollierungsparametern **80**
Konfigurieren von Prozess- und Dienstparametern **75**
Konfigurieren von Sandbox-Parametern **96**
Konfigurieren von Versionsparametern **81**
LocaleIdentifier **82**
LocaleName **82**
LogPath **80**
MSIArpProductIcon **91**
MSIDefaultInstallAllUsers **92**
MSIFilename **92**
MSIInstallDirectory **93**
MSIManufacturer **93**
MSI-Parameter **33**
MSI-Parameter ändern **34**
MSIProductCode **93**
MSIProductVersion **94**
MSIRequireElevatedPrivileges **94**
MSIUpgradeCode **95**
MSIUseCabs **95**
NetRelaunch **58**
NotificationDLLs **70**
NotificationDLLSignature **70**
ObjectTypes **71**
OptimizeFor **79**
OptionalAppLinks **88**
OutDir **64**
Parameter, die für die ##Attributes.ini-Datei gelten **58**
PermittedGroups **66**
Protocols **63**
ReadOnlyData **84**
RegistryIsolationMode **62**
RemoveSandboxOnExit **97**
RequiredAppLinks **87**
ReserveExtraAddressSpace **84**
RetainAllIcons **64**
RuntimeEULA **59**
SandboxCOMObjects **71**
SandboxName **97**
SandboxNetworkDrives **97**
SandboxPath **98**
SandboxRemovableDisk **98**
Shortcut **84**
Shortcuts **85**
Source **85**
StripVersionInfo **81**
Struktur **58**
UACRequestedPrivilegesLevel **67**
UACRequestedPrivilegesUiAccess **67**
UpgradePath **73**
Version.XXXX **81**
VirtualComputerName **59**
VirtualDrives **73**
VirtualizeExternalOutOfProcessCOM **71**
WorkingDirectory **86**
Wow64 **60**
Package.ini-Parameter **57**–**98**
Pakete
definieren **21**
nach dem Kapselungsvorgang konfigurieren **24**
Paketieren von Microsoft Office **25**–**28**
Parameter
Einstellungen auf Ordnerebene statt auf Paketebene anwenden **25**
für MSI-Dateien **33**
für sbmerge.exe **53**
für thinreg.exe **31**
PermittedGroups, Auswirkung auf Application Link **50**
Personal Edition
ausgeschlossene Funktionen **9**
primärer Datencontainer
Beibehalten des Namens mit Application Sync **45**
definieren **22**
Größenauswirkungen **22**
Problembehandlung
Bereitstellung der benötigten Informationen für den Support **123**
Explorer.exe **132**
Java Runtime Environment **133**
Microsoft Outlook **131**
mit Protokoll-Monitor **124**
Projektdateien **23**
Projekte
während des Kapselungsvorgangs öffnen **23**
Protokollformat **126**
Protokoll-Monitor
Anhalten und Fortsetzen der Protokollierung **125**
Problembehandlungsverfahren **124**
verwenden **124**

zusätzliche Optionen **125**

R

regedit.exe, definieren **17**
Relink
Beispiele **55**
definieren **55**

S

Sandbox
definieren **20**
Parameter **96**
Speicherort **21**, **101**
Struktur **102**
Suchreihenfolge **99**
Überlegungen für aktualisierte Anwendungen **55**
sbmerge.exe
Befehle **53**
definieren **51**
Zusammenführen der Laufzeitänderungen **52**
Setup Capture-Assistent
Anwendungen installieren **16**
Benutzergruppen autorisieren **18**
Bestandsname **21**
Einstiegspunkte **17**
Festlegen der Isolationsmodi **20**
Nachüberprüfungsvorgang **16**
Pakete erstellen **23**
Paketeinstellungen **22**
Projekte durchsuchen **23**
Projektspeicherort **21**
Vorprüfungsvorgang **16**
Setup Capture-Assistent, verwenden **16**–**24**
Shell-Integration, beschränkte Funktionen mit ThinApp **12**
Skripts
Beispiel: .bat **112**
Beispiel: .reg **113**
Beispiel: Dateikopie **114**
Beispiel: Dienst **113**
Beispiel: Systemregistrierung **114**
Beispiel: Timeout **113**
Beispiel: Virtuelle Registrierung **113**
Gründe für **112**
Rückruffunktionen **111**
snapshot.exe
Beispielbefehle **105**
Beispielvorgang **105**
Erstellen der Snapshots von der Befehlszeile **103**
snapshot.ini, definieren **103**, **106**

T

Technischer Support
erforderliche Angaben für die Problembehandlung **123**
ThinApp
Aktualisieren der Laufzeit in Paketen **55**
Aktualisieren von Anwendungen **43**
Anforderungen zur Installation und Kapselung von Anwendungen **11**
Bereitstellungsoptionen **29**
Empfehlungen für sauberen Computer **12**
in einer VMware View-Umgebung **29**
nicht unterstützte Anwendungen **12**
Ordnermakros **107**
Pakete per Stream vom Netzwerk übertragen **37**
Projektdateien durchsuchen **23**
thinreg.exe verwenden **30**
unterstützte Betriebssysteme und Anwendungen **11**
Verzeichnisdateien **13**
wird installiert **13**
thinreg.exe
Aktualisieren der Registrierungen mit Application Sync **45**
ausführen **31**
definieren **30**
mit Application Sync **31**
Parameter **31**
Starten mit MSI-Dateien **22**
Treiber, Unterstützung für **39**

U

Unterstützung
für Anwendungen **11**
für Betriebssysteme **11**

V

verschachtelte Links mit Application Link **49**
virtuelles Dateisystem
Formatierungsstufen **107**
Pfade mit Makros darstellen **107**
verwenden **107**
VMware View, unter Verwendung von gekapselten Anwendungen **29**
vregtool, Auflisten virtueller Registrierungsinhalte **102**